UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior

de

Ingeniería y Sistemas de Telecomunicación

PROYECTO FIN DE CARRERA

Aplicación Android para el aprendizaje de inglés nivel B2. Extensión del ILLLab

Gerardo Héctor Glorioso

SEPTIEMBRE 2015

A mis padres, por haber confiado en mí y haberme apoyado todos estos años

A mi tutora Irina por haberme animado y apoyado para seguir este proyecto

RESUMEN

Desde el inicio de la globalización, el aprendizaje de la lengua inglesa se ha instaurado como una necesidad. Hoy en día, con la adopción del Espacio Europeo de Educación Superior este lenguaje no sólo se impone como un requisito para los estudiantes sino que se exige un nivel B2, lo cual significa un esfuerzo mayor tanto como para el alumno como para el profesor a la hora de hacer de este ejercicio un hábito y lograr la evaluación continua de los mismos.
Este proyecto intenta extender las funcionalidades de una aplicación existente llamada Illlab con ejercicios que se adapten al nivel B2 y permitan la interacción entre alumnos durante la realización de estos ejercicios.

El objetivo de esta aplicación es el de desarrollar ejercicios extra en la aplicación Illlab que añadan complejidad para el aprendizaje de inglés de un nivel B2 y que además se puedan realizar actividades entre los alumnos. La idea es hacer una aplicación de preguntas y respuestas "multiple choice" con cuatro opciones por pregunta. El fuerte de este juego está en presentar material variado sobre uso de la lengua y además permitir el juego entre varios alumnos.

La extensión de ILLLab se plantea como un proyecto para desarrollar interfaces y funcionalidades adicionales en la antigua aplicación. La principal funcionalidad que se añade es un juego de preguntas y respuestas con opciones múltiples para un nivel B2 y las interfaces responden a necesidades de intercambio y manejo de contenido por Internet mediante estándares aceptados en el mundo del aprendizaje digital tales como Common Cartridge o SCORM.

Este proyecto simplemente adapta la aplicación para su uso en un entorno de evaluación de actividades en el cual el profesor tiene acceso a las actividades que realizan los alumnos de un curso para su posterior evaluación. Antiguamente ILLLab sólo contenía ejercicios que se llevaban a cabo en el dispositivo móvil por lo que el control de estas actividades no era posible.
La mejora se presenta como una interfaz Common Cartridge para el manejo del contenido, una interfaz de comunicación sobre servicios web tipo REST y el manejo de base de datos mediante Hibernate que agrupa una serie de librerías Java para la persistencia de objetos de la base de datos.

ABSTRACT

Since the onset of globalization, the learning of the English language has become as a necessity. Today, with the adoption of the European Higher Education Area this language is not only imposed as a requirement for students but a B2 level is required, which means a greater effort both to the student and teacher when it comes to make the learning exercise a habit and achieve continuous evaluation of students.

This project aims to extend the functionality of an existing application called Illlab with an exercise that suits the B2 level and allow interaction between students while performing these exercises.

The purpose of this application is to develop an additional exercise in the application Illlab that adds complexity for learning English at B2 level and also enables the interaction among students. The main idea is to make an application in multiple choices style with four options. The strength of this game is to present varied material on use of Enlgish and also allow play between two students.

ILLLab extension is conceived as a project to develop interfaces and additional functionalities in the old application. The main functionalities added are a game of questions and answers with multiple choices for a B2 level and interfaces that meet information exchange requirements and content management over the Internet using standards adopted in the world of digital learning such as Common Cartridge or SCORM.

This project simply adapts the application for its use in an activities evaluation environment in which the teacher has access to the activities performed by students in a course for further evaluation. The former versión of ILLLab contained only exercises that were carried out on the mobile device so that the evaluation of these activities was not possible.

The improvement comes as a Common Cartridge interface for content management, a communication interface with REST web services and a database access using Hibernate which groups a number of Java libraries for object persistence in the database.

Índice

Glosario

ILLLab	Integrated Language Learning Lab
Android	Sistema operativo para dispositivos móviles
DMA	Desarrollo rápido de aplicaciones
SDK	Software Development Kit
PFC	Proyecto de Fin de Carrera
UML	Unified Modelling Language
API	Application Programming Languaje
SO	Sistema Operativo
SQL	Structured Query Language
SCORM	Sharable Content Object Reference Model
LMS	Learning Management System
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics
MVC	Modelo Vista Controlador
Common Cartridge	Formato de información que cumple las especificaciones de empaquetado de contenido y normal de interoperabilidad establecidas en el IMS Global Learning Consortium
MySQL	Sistema de gestión de bases de datos relacional
MySQLWorkbench	Herramienta visual de diseño de bases de datos que integra desarrollo de software, Administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL
eXe	Software de edición de contenidos
PC	Personal Computer
TCP	Transmission Control Protocol
Tomcat	Servidor web de aplicaciones Java
FTP	File Transfer Protocol
Java	Lenguaje de programación orientado a objetos
C	Lenguaje de programación de propósito general
XML	eXtensive Markup Language
GUI	Graphical User Interface
Web Service	Servicio Web, tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones

Json	JavaScrip Object Notation. Formato Ligero para el intercambio de datos
HTML	Hyper Text Markup Language
REST	Arquitectura software para sistemas hipermedia distribuidos como la World Wide Web
CD	Compact Disc
Hibernate	Herramienta de Mapeo Objeto-Relacional (ORM) para la plataforma Java
EER	Enhanced entity-relationship model
USB	Universal Serial Bus
JavaDoc	Utilidad de Oracle para la generación de documentación de APIs en formato HTML a partir de código fuente Java
EDP	Estructura de Descomposición el Proyecto
COCOMO	Modelo Constructivo de Costos
Gantt	Diagrama de Gantt. Se utiliza para reflejar tareas y su extensión en el tiempo
LMS	Learning Management System

Lista de ilustraciones

Lista de tablas

1 INTRODUCCIÓN Y OBJETIVOS

La extensión del ILLLab que se presenta en este proyecto tiene como objetivo integrar ejercicios para el aprendizaje de inglés de un nivel B2 y que además se puedan realizar actividades entre los alumnos de una clase. La idea que se presenta es una aplicación de preguntas y respuestas de opción múltiple (multiple choice) con cuatro opciones de respuesta por pregunta.

Esta aplicación se realizó siguiendo una metodología de trabajo en espiral, lo que significa que se han realizados las actividades de captura de requerimientos, diseño y desarrollo de una forma iterativa para que la especificación final de la aplicación tenga una mayor consistencia. De esta forma, los objetivos conseguidos junto con la descripción de la base teórica utilizada, se muestran en cada fase de desarrollo y son presentados en los siguientes capítulos.

El principal objetivo de este proyecto es el desarrollo de la extensión de ILLLab. Esto se puede separar en distintos sub objetivos derivados del desarrollo, que son los que se presentan en cada capítulo:

Base teórica:

- Comparativa de metodologías
- Características del proyecto y elección de metodología
- Base teórica, tecnologías y estándares utilizados en cada fase

Desarrollo:

- Comunicación: Lista de casos de uso y requerimientos
- Diseño de la aplicación junto con la herramienta de diseño y modelo utilizado.
- Código y despliegue de datos. Tecnologías y herramientas elegidas, configuración, código fuente, scripts de bases de datos y contenido.

Validación

- Tareas realizadas y resultados contrastados con requerimientos

Presupuesto

- Esfuerzo, costes de desarrollo y amortización

En el primer capítulo de esta memoria se presenta la base teórica sobre la que se ha desarrollado la aplicación. Aquí se muestran los conceptos de metodología de desarrollo del proyecto, sus fases y las técnicas que se emplean en cada una. En el segundo capítulo se presenta el desarrollo de la

aplicación acorde a las fases indicadas en la metodología de trabajo que incluye la lista de requerimientos, el diseño y el código. Además, se exponen las herramientas que se han utilizado indicando las configuraciones necesarias. Seguido, en el tercer capítulo, se describe la fase de validación en dispositivos reales para comprobar todas las funcionalidades y la usabilidad de la aplicación.

El cuarto capítulo comprende un análisis del presupuesto que se requiere para el desarrollo de la extensión de ILLLab. Esto incluye el esfuerzo requerido y costes derivados de esta actividad tales como el uso de herramientas de pago, instrumentos especiales que se requieren, etc. Además, se añade una pequeña discusión sobre los modelos de negocio y la forma de amortizar el gasto del desarrollo.

El último capítulo es el de las conclusiones. Aquí se da espacio para reflexiones con respecto a la metodología elegida, tecnologías para el desarrollo, comentarios sobre el diseño, el código fuente y las herramientas utilizadas. Más allá de los comentarios sobre el desarrollo en general, también se proponen futuras líneas de trabajo para plantear una homogeneización de la aplicación y su despliegue en entornos de producción.

2 BASE TEÓRICA

2.1 COMPARATIVA DE METODOLOGÍAS

Antes de la selección de una metodología es necesario ubicarse en el contexto en el que se intenta desarrollar este proyecto. Debido a que este es un proyecto de final de carrera, el cliente es sólo la persona que da los requisitos funcionales necesarios que debe cumplir la aplicación. En este caso es el/la profesor/a de lingüística. El ingeniero de software es el alumno y es el que se encargará de realizar todas las fases de desarrollo.

Según [1], existen distintos modelos dependiendo de cómo sea la naturaleza del proyecto a realizar:

1. Modelos en cascada
2. Modelos incrementales
3. Modelos evolutivos

Dentro de éstos modelos existe procesos comunes a todos que se han adoptado como un marco de trabajo tradicional de desarrollo de software. Estos procesos comprenden las actividades de (1) comunicación, (2) planificación, (3) modelado, (4) construcción y (5) despliegue. En la fase de comunicación se recopilan los requerimientos, en la de planificación se hace estimaciones de recursos y tiempo para el desarrollo, en la fase de modelado se realiza un análisis y representación del software, en la de construcción de codifica el software y se realizan pruebas y por último en la fase de despliegue se hace la entrega del proyecto y se da soporte para el mantenimiento.

Dependiendo de cómo sea la naturaleza del proyecto se deben seguir las actividades de forma lineal, iterativa, etc. Por ejemplo, en el modelo en cascada se realizan las tareas de forma lineal debido a que los proyectos a los que se aplica son conocidos por el grupo de trabajo. Esto suele pasar en procesos de mejoras del software. Sin embargo, el modelo en cascada no es del todo realista porque no da lugar a error entre las fases de desarrollo.

En el modelo incremental se plantean las mismas fases pero, en cambio, se da lugar a repetir las fases desde el principio desde la base de la que se partió en la iteración anterior. Los proyectos que responden a este modelo suelen ser de mejoras incrementales o en las que sea necesario realizar entregas rápidas y luego ir refinando y expandiendo funcionalidades. Aunque se puede volver a etapas anteriores, en este modelo se plantea el desarrollo de las fases en cascada en cada iteración.

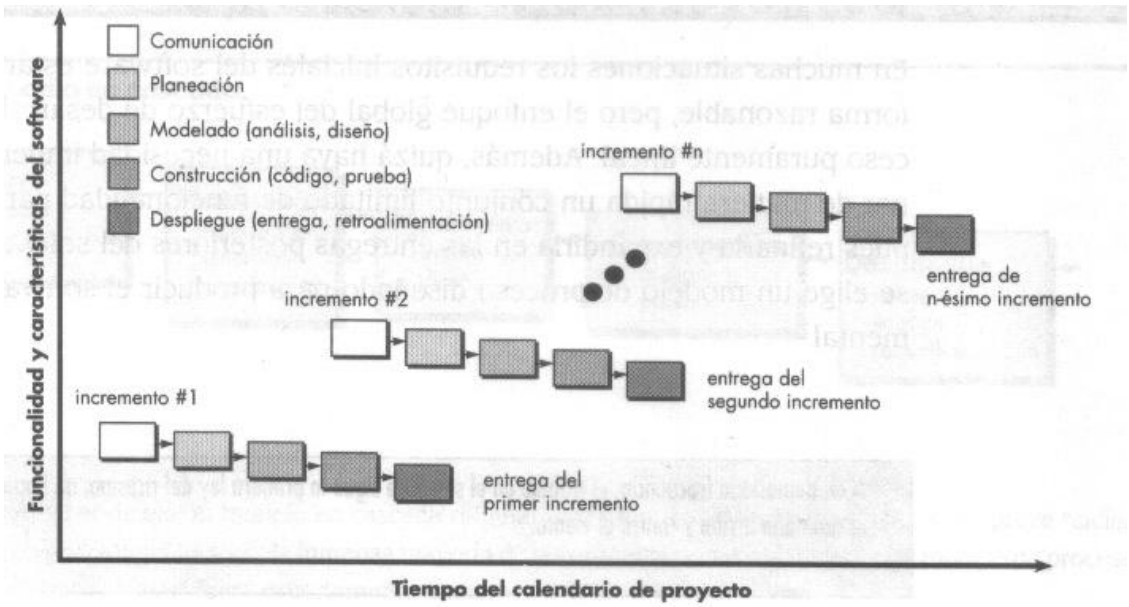


ILUSTRACIÓN 1 MODELO INCREMENTAL[1]

Existe una variante del modelo incremental que se incluye en la misma categoría (modelos incrementales). Este modelo lleva el nombre de "desarrollo rápido de aplicaciones"(DMA). La diferencia con el modelo anterior es que en este caso el desarrollo es llevado a cabo por muchos equipos que aplican individualmente por grupos la metodología en cascada. Cada equipo trabaja sobre diferentes funcionalidades del sistema por lo que el tipo de proyectos a abordar con esta metodología suelen requerir recortar considerablemente el tiempo de desarrollo.

Aun adoptando el tipo de modelos iterativos, es complicado adaptarse a mundo de los negocios donde los requerimientos cambian con frecuencia. Por lo tanto, es necesario aplicar otro tipo de metodologías que permitan adaptarse a estos cambios. Este tipo de modelos entran en la categoría de los modelos evolutivos que son iterativos pero que permiten presentar versiones mejoradas en cada iteración.

Una implementación de este modelo es la construcción de prototipos. Esta metodología permite realizar versiones y presentar prototipos al cliente para redefinir funcionalidades o requerimientos. Sin embargo, este modelo tiene el problema de que el cliente ve en el prototipo una solución final al problema y suele ordenar el cierre del proyecto sin haber tenido en cuenta la calidad final del software o un plan de mantenimiento sólido. Además, al intentar realizar versiones que funcionen rápidamente, se hacen elecciones sobre sistemas operativos o lenguajes de programación que no son los más apropiados para el tipo de software que se intenta desarrollar y se terminan adoptando para seguir la dinámica de trabajo.

Otra alternativa a la construcción de prototipos es el modelo en espiral en el que las fases de comunicación, desarrollo, etc. se realizan en paralelo. Por lo tanto en cada iteración de cada fase se

pueden aprovechar a realizar cambios significantes al software y así presentar diferentes prototipos con mejoras.

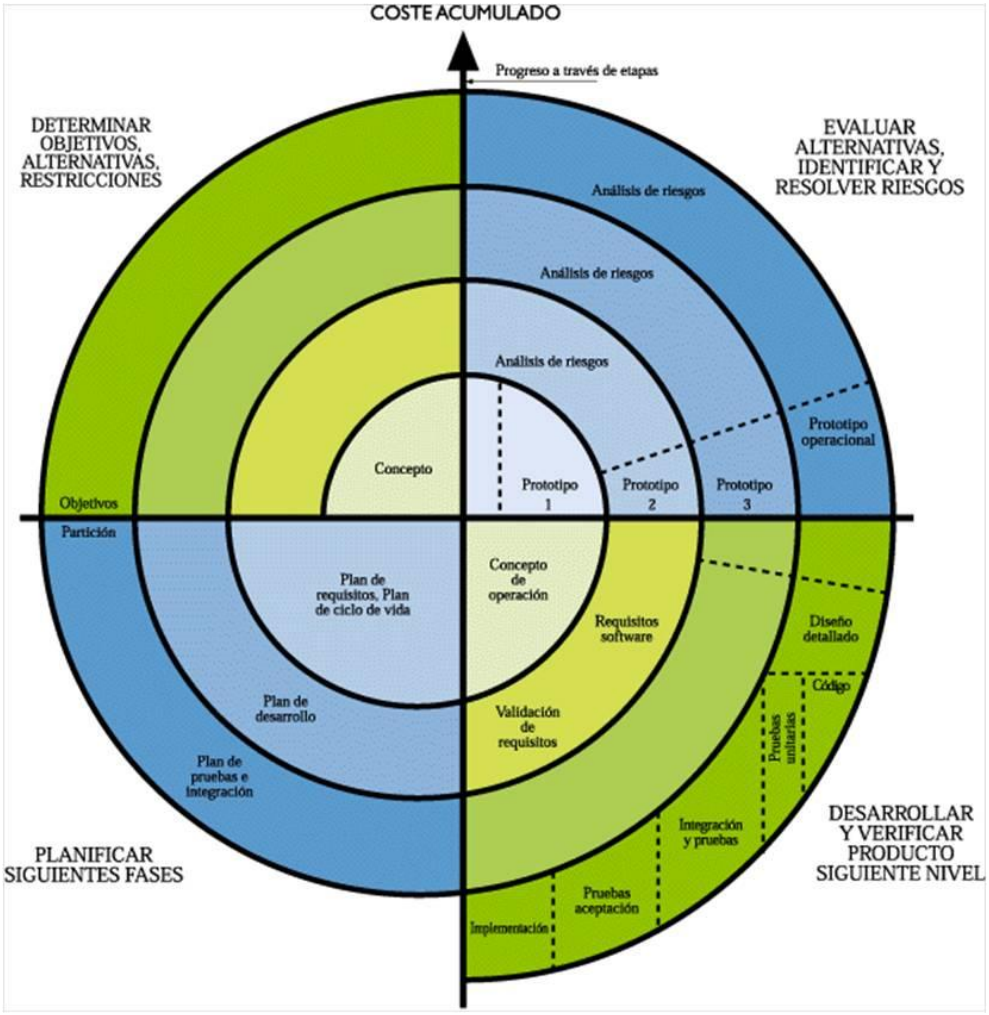


ILUSTRACIÓN 2 MODELO EN ESPIRAL[2]

Si has usado alguna bibliografía para el apartado 1.1. cítala en el cuerpo del texto.

2.2 CARACTERÍSTICAS DEL PROYECTO Y ELECCIÓN DE METODOLOGÍA

Tal como se ha explicado anteriormente, las metodologías responden a necesidades de los ingenieros de software con el objetivo de realizar una gestión eficiente de los recursos y el tiempo. No obstante, hay tres aspectos principales a considerar en la elección de una metodología:

- La complejidad del problema
- Los recursos con los que se cuentan

- La interacción con el cliente y su perfil

En este caso, la complejidad del problema se presenta mayormente en el desarrollo de la aplicación, por lo tanto no es necesario volver con frecuencia a la redefinición de requerimientos y conceptos pero sí es necesario comprobar que lo que se desarrolla cumple con las funcionalidades requeridas.

Los recursos con los que se cuentan en este proyecto son escasos. No hay inversión para la compra de materiales y se debe desarrollar con lo que el alumno disponga. En concreto, el proyecto sólo necesita de un ordenador para el desarrollo de la aplicación y dispositivos móviles para realizar pruebas. En caso de que no se disponga de un móvil, se puede probar en alguno prestado o se puede subir la aplicación al mercado de Google para que los usuarios den comentarios al desarrollador.

El cliente es el profesor de inglés de la Universidad. Su disposición para entrevistas de toma de requerimientos es como la de cualquier profesor por lo que se pueden realizar las tutorías una o dos veces por semana. El perfil del cliente no es tecnológico por lo que el lenguaje es fácil de interpretar pero da un poco de lugar a la ambigüedad según qué terminología se utilice.

Teniendo en cuenta esos aspectos, se puede caracterizar el proyecto como de complejidad media, de bajos recursos y fácil interacción con el cliente. Sin embargo, es necesario realizar todas las fases de desarrollo e incluso volver en caso de que algún concepto o requerimiento no esté claro.

El desarrollo de la aplicación es de mejora o extensión de funcionalidades de ILLLab, por lo que un proceso iterativo es necesario. También es necesario hacer entregas rápidas para que el cliente visualice las funcionalidades que se quieren conseguir más fácilmente.

En conclusión, es necesario aplicar una metodología incremental debido a que es preciso hacer entregas relativamente rápidas, el desarrollo no lo realizan múltiples equipos y se pueden volver a definir requerimientos o desarrollar nuevas funcionalidades.

2.3 BASE TEÓRICA, TECNOLOGÍAS Y ESTÁNDARES UTILIZADOS EN CADA FASE

2.3.1 COMUNICACIÓN

En esta fase del desarrollo se realiza la comunicación con el cliente para la generación de requerimientos que debe cumplir el software. Generalmente, se realiza una entrevista con el cliente mediante la cual se negocian las funcionalidades del software que se pueden realizar, las que no son factibles y las que son necesarias. Al concluir con las entrevistas se genera un documento de trazabilidad de requerimientos para luego validar todas las funcionalidades que se han acordado.

En este proyecto se realizaron varias entrevistas para definir los casos de uso y limitar hasta dónde se quiere llegar con la aplicación. El lenguaje para representar los casos de uso fue UML que ofrece una gran variedad de gráficos para el diseño de software[3]. Una vez definidos los casos de uso se hizo un mock-up para representar de forma visual lo que se había hablado durante las entrevistas.

El mock-up se incluyo para comprobar que lo que el desarrollador entiende concuerda con las expectativas del cliente (profesor/tutor). Según el informe técnico ISO/IEC TR 14759, un mock-up se diferencia de un prototipo en que el mock-up no llega a mostrar funcionalidades ni puede operar en un entorno real sino que es más una forma de ilustrar el software que se quiere conseguir.

La herramienta de mock-up que se utilizó se llama "MIT app Inventor" de Google. Esta herramienta permite crear un projecto y diseñar varias pantallas mediante recursos similares a las que presenta el (Software Development Kit) de Android para Eclipse: botones, etiquetas, etc.

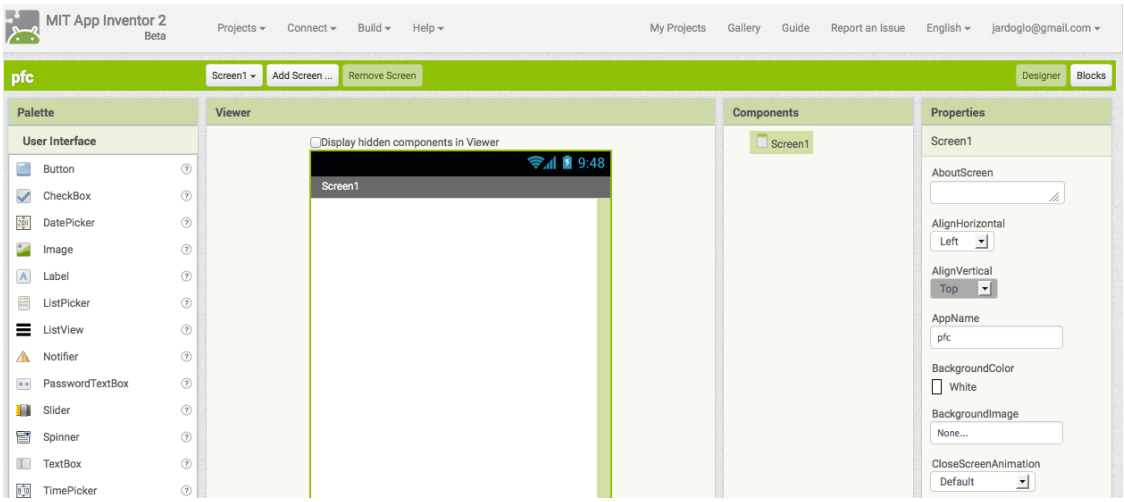


ILUSTRACIÓN 3. HERRAMIENTA MIT APP INVENTOR DE GOOGLE

Finalmente, una vez definidos los casos de uso, se realizó una tabla de requerimientos para comprobar al final de la presentación de un prototipo que las funcionalidades definidas se han implementado. Esta tabla incluye requerimientos funcionales y no funcionales. Los requerimientos funcionales son las prestaciones que la aplicación debe dar al usuario final y cómo debe funcionar. Los requerimientos no funcionales en cambio representan barreras tecnológicas o imposiciones para el desarrollo de la aplicación tales como sistema operativo, lenguaje de programación, herramientas a utilizar, protocolos, estándares a tener en cuenta, etc.

2.3.2 PLANIFICACIÓN

Debido a la naturaleza de este proyecto (PFC), las actividades a realizar sólo se pueden ver afectadas por factores exteriores que no suelen ser tenidos en cuenta para la definición de riesgos en proyectos de ingeniería, por lo tanto la planificación se limitó a la realización de un presupuesto y estimación de herramientas y horas de trabajo.

2.3.3 MODELADO

Un modelo es una representación de algo que se quiere construir. Cuando el modelo es de un objeto físico, se puede construir con materiales y crear un objeto en otra escala. Cuando el objeto es software, el modelo debe representar su estructura y comportamiento con un lenguaje específico que lo permita.

El modelado de la aplicación se llevó a cabo en UML (Unified Modelling Language) [4]. Este lenguaje incluye una serie de gráficos que permiten la creación de modelos de software desde su arquitectura (estructura) hasta la lógica de sus procesos e intercambio de información (comportamiento).

En el caso del software, se deben representar dos modelos, (1) el modelo del análisis y el (2) modelo del diseño en el que (1) define la estructura y componentes que interaccionan en el sistema y (2) entra en la especificación de la implementación del sistema [5]. En este proyecto se han elaborado ambos modelos y en cada uno se han representado los siguientes sub-modelos:

- (1)Arquitectura
- (1)Modelo del dominio de la información
- (1)Modelo del dominio funcional
- (1)Modelo del dominio de interacción con el usuario
- (2) Modelos de diseño: diagramas de secuencia, actividad, etc.

2.3.4 CONSTRUCCIÓN

El proceso de construcción del software comprende la codificación y pruebas. En este caso se desarrolló la aplicación en el lenguaje de programación Java. Se eligió este lenguaje porque tiene un SDK bastante avanzado y porque la integración con la primera versión de ILLLab es inmediata.

Existen varias herramientas para realizar proyectos en Java tales como NetBeans [6] o Eclipse [7]. Cualquier de las dos herramientas sirve para el desarrollo de la aplicación así que en este caso se eligió Eclipse simplemente por la facilidad de uso y familiaridad que el desarrollador tiene con esta herramienta.

Android tiene una serie de APIs (Application Programming Interface)[8] integrados en el SDK que permite programar fácilmente las aplicaciones móviles sin tener que acceder a funciones del SO (sistema operativo). Además de las APIs, el SDK también ofrece una interfaz de diseño gráfico que genera código fuente al ir integrando componentes gráficos a la interfaz de usuario. El SDK de Android esta disponible para varias versiones del sistema operativo[9].

Esta aplicación, además de tener el componente de interfaz gráfica de Android, tiene dos componentes para el acceso a datos y al contenido respectivamente. Las tecnologías elegidas para la comunicación y acceso a datos se discuten en la sección de desarrollo porque son parte de las librerías que proporciona Java. Sin embargo es necesario definir el tipo de base de datos y estándares de metadatos sobre el contenido que se han elegido.

Para la implementación de la base de datos se ha elegido el modelo de base de datos relacional debido a que es necesario que sus objetos tengan referencias hacia otros y así mantener consistencia en la creación de sus instancias y evitar crear información redundante. El lenguaje de consulta elegido es SQL[10](Structured Query Language) y la herramienta de administración y modelado de la base de datos es el software MySQL Workbench[11].

Uno de los impedimentos en la reutilización de código e interoperabilidad entre aplicaciones es el acceso a los datos de una forma estándar. Actualmente existen dos estándares para compartir datos en el contexto de elearning, estos son los estándares SCORM y Common Cartridge. Sin embargo el alcance de ambos es distinto desde el punto de vista de qué tipo de información se quiere gestionar[12]. El estándar SCORM es para información que esta diseñada para ser leída por el alumno y en la que sólo interacciona con el ordenador. Common Cartridge, en cambio, esta pensado para lo que se llama “blended-learning” donde se diseña un curso por el profesor o comunidad y el alumno realiza ejercicios.

Evaluando las características de los dos estándares, Common Cartridge es el más adecuado para implementar la interfaz de intercambio de datos entre el LMS(Learning Management System) de ILLLab y la interfaz de consumo de datos de la aplicación. El LMS consta de la herramienta de gestión de bases de datos presentada anteriormente y otra herramienta llamada Exe que es un editor de ejercicios de tipo “multiple choice” que permite exportar información en formato Common Cartridge[13]

Por último, las pruebas realizadas se detallan en el capítulo 4

3 DESARROLLO

3.1 COMUNICACIÓN: LISTA DE CASOS DE USO Y REQUERIMIENTOS

3.1.1 INFORME DE ENTREVISTA PARA GENERACIÓN DE REQUERIMIENTOS. DESCRIPCIÓN DE APLICACIÓN ANDROID INGLÉS B2

La entrevista se realizó directamente con el departamento de Lingüística de la Universidad Politécnica de Madrid. Se planteó el funcionamiento que la aplicación debe tener y cómo se debe gestionar.

En principio, la idea es hacer una aplicación de preguntas y respuestas "multiple choice" con cuatro opciones para practicar el inglés. El material se centrará en ejercicios de "use of English" y el nivel debe ser B2 (para titulación de First Certificate).

FUNCIONAMIENTO DE LA APLICACIÓN

Esta aplicación debe mostrar al usuario, en la pantalla principal, dos modalidades de juego:

1. Un jugador: Modalidad normal con tiempo en cada pregunta. Al final se da la calificación basada en el tiempo y en las preguntas correctas.
2. Dos jugadores: Un jugador invita a otro. El otro acepta o rechaza la partida.

En la modalidad de un jugador se accede a un menú de opciones en el que se da la oportunidad de elegir qué tipo de modalidad quiere jugar. Seguido de la elección, la aplicación dará al usuario una serie de preguntas con múltiples respuestas donde el usuario debe seleccionar la correcta dentro de un rango determinado de tiempo.

En cada pregunta aparecerá la pregunta en sí o una oración en la que haya que insertar la opción correcta y una barra de tiempo para añadir dificultad.

El criterio de evaluación es de 1 a 10, sin embargo el tiempo influye en la calificación. De esta forma, sólo se cuenta que la pregunta sea correcta o no y luego, en caso de que sea correcta la puntuación dependerá del tiempo que se haya tardado en responderla.

En caso de que el jugador salga de la aplicación en medio de una partida se guardará automáticamente la última partida en la que estaba para continuar la próxima vez que entre.

Cuando el jugador entre nuevamente a la misma partida se le pregunta si quiere continuar la última partida y en caso de que no quiera se borra y comienza una nueva.

En la modalidad de dos jugadores se juega asíncronamente. Para iniciar una partida un jugador debe invitar a otro y una vez que acepte se inicia la partida. El juego se lleva a cabo como si fuera para un solo jugador.

Una vez que uno ha terminado la partida se le enseña a ambos jugadores su puntuación. Cuando ambos jugadores terminan se muestra el ganador.

Cada juego tendrá un inicio donde se explica el tiempo que tiene el jugador para responder cada ejercicio y un botón de confirmación parar jugar.

Una pantalla de login se debe incluir al inicio de la aplicación para acceder debido a que el uso debe estar restringido a estudiantes de la Universidad Politecnica de Madrid o a la institución que se esté gestionando.

3.1.2 CASOS DE USO

Los casos de uso son diagramas que indican la interacción del sistema (en recuadro) con un actor externo, en este caso, los usuarios finales que son el alumno y el profesor. Este diagrama permite ver a grandes rasgos las funcionalidades que la aplicación ofrece. Cada una de estas funcionalidades están indicadas en círculos dentro del diagrama.

Para generar los gráficos de los casos de uso, UML ofrece el paquete de gráficos de casos de uso que incluye las funcionalidades, actores y relaciones entre casos de uso. En concreto, se ha utilizado la herramienta TopCoder UML Tool para Mac[14]

Generación de casos de uso con TopCoder UML Tool

Para crear un fichero de gráficos en esta herramienta simplemente se va a File->New y se selecciona el lenguaje de programación que se va a utilizar; en este caso Java.

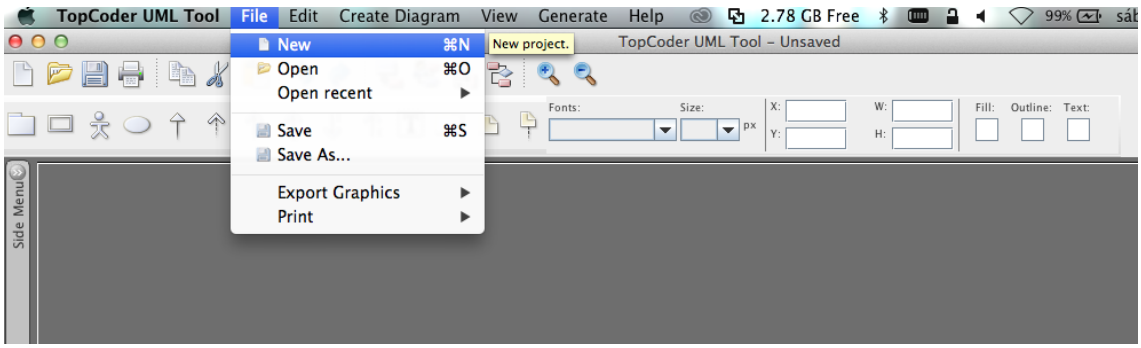


ILUSTRACIÓN 4 GENERACION DE CASOS DE USO CON TOPCODER: OPCION 1

Una vez creado el proyecto se selecciona “Create New Use Case Diagram” de la barra de herramientas para crear una plantilla.

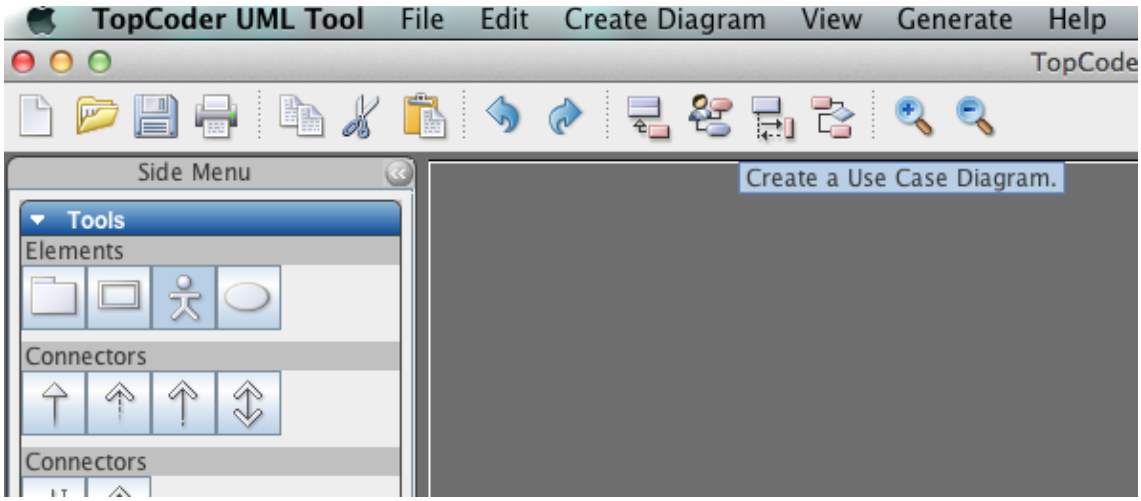


ILUSTRACIÓN 5 GENERACION DE CASOS DE USO CON TOPCODER: OPCION 2

Para generar el gráfico simplemente se desplazan los elementos a la plantilla. Una vez generado el gráfico se puede exportar en formato JPEG, PNG, etc. Esto se hace yendo a File->Export Graphics->Export Current Diagram.

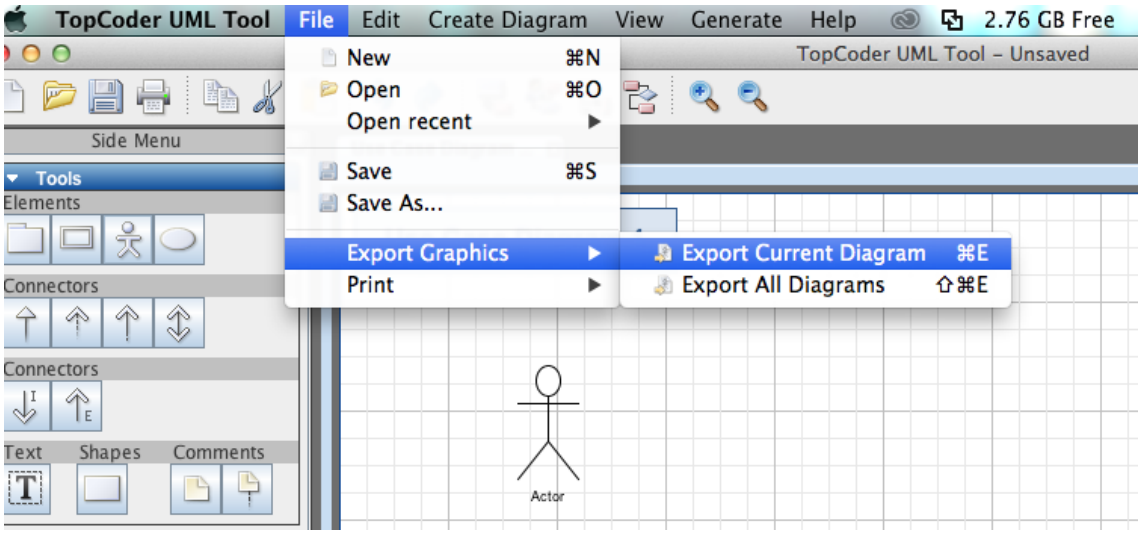


ILUSTRACIÓN 6 GENERACION DE CASOS DE USO CON TOPCODER: OPCION 3

Relaciones entre casos de uso

La etiqueta “include” que se ve en el diagrama indica que el caso de uso apuntado es parte del que esta unido al mismo. Por ejemplo, “Ver sesiones” implica tener que actualizar las sesiones existentes y al mismo tiempo es parte de la gestión de sesiones.

La etiqueta “extends” indica que un caso de uso es complementario a otro, es decir, son parte de la misma funcionalidad pero particularizados para un caso. En este diagrama se ve que Jugar Partida se puede hacer individualmente pero también es parte de Jugar Partida Sesión que es cuando un jugador juega contra otro.

Casos de uso de la extensión de ILLLab

El siguiente diagrama ilustra la definición de casos de uso. Luego se explica cada uno para definir en qué consiste para a continuación generar una serie de requerimientos generales que debe cumplir la aplicación.

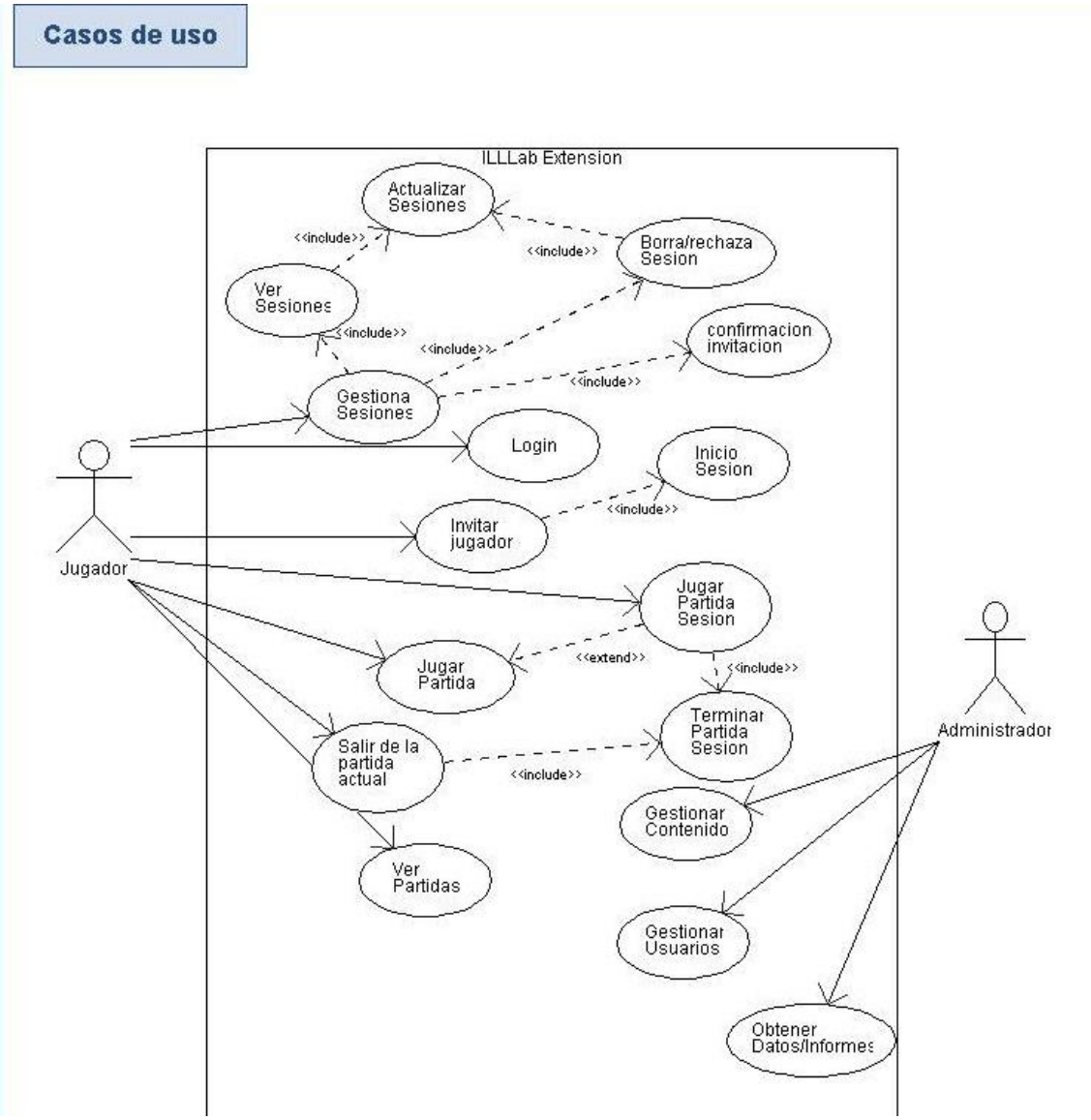


ILUSTRACIÓN 7 DIAGRAMA DE CASOS DE USO DE ILLLAB

Definición de casos de uso

TABLA 1 DEFINICION DE CASOS DE USO

[nº-rf-codigo]	Nombre	Descripción
1-cu-GS	Gestiona sesiones	El jugador crear o rechazar una sesión y visualizar un grupo de sesiones
2-cu- BRS	Borra/Rechaza sesión	El jugador rechaza una sesión a la que ha sido invitado
3-cu- AS	Actualizar sesiones	Esto es parte de la aplicación. Cada vez que el jugador quiere ver las sesiones activas o terminadas, se actualizan en una lista
4-cu- VS	Ver sesión	En este caso el usuario (alumno) ve las sesiones que ha generado con otro jugador, en las que ha sido invitado, las sesiones terminadas y las que están en juego
5-cu-CI	Confirmar invitación	El jugador confirma una invitación a jugar una sesión
6-cu- LI	Log in	El jugador se autentica ante la aplicación para tener acceso al contenido
7-cu- IJ	Invitar jugador	Un jugador puede invitar a otro a jugar de una lista de jugadores disponibles
8-cu- IS	Inicio sesión	Una sesión se inicia al invitar a otro jugador
9-cu- JP	Jugar partida	El jugador juega una partida individual y se guardan los resultados
10-cu-JPS	Jugar partida sesión	El jugador inicia una partida de una sesión en juego siempre y cuando no haya jugado
11-cu-SPA	Salir de la partida actual	El jugador sale de la partida actual y sus datos son guardados junto con el cómputo de la puntuación aunque no haya terminado la partida
12-cu-VP	Ver partidas	El jugador visualiza las partidas individuales jugadas
13-cu-GC	Gestionar contenido	El administrador del contenido puede agregar, borrar y actualizar contenido para los ejercicios

14-cu-GJ	Gestionar jugadores	El administrador agrega y borra jugadores de la base de datos
15-cu-ODI	Obtener datos e informes	El administrador genera listados de partidas jugadas, sesiones creadas y sus estados.

3.1.3 ESPECIFICACIÓN DE REQUERIMIENTOS

Los casos de uso definen el uso que debe presentar la aplicación. Luego se deben definir los requerimientos funcionales y no funcionales para dar una idea al desarrollador de los límites de las funcionalidades y las imposiciones de carácter tecnológico (hardware, diseño, software, etc.).

Requerimientos Funcionales

TABLA 2 REQUERIMIENTOS FUNCIONALES

[nº-rf-codigo]	Descripción	Métrica
1-rf-gui	El interfaz gráfico debe permitir elegir entre único jugador y dos jugadores	-
2-rf-gui	El usuario puede elegir el tema sobre el que se va a evaluar: gramática, vocabulario o uso del inglés	-
3-rf-gui	El usuario puede empezar con la última partida guardada o con otra. Sólo se guarda la última partida en modo un solo jugador.	-
4-rf-dif	El juego no ofrece distintos niveles de dificultad	-

5-rf-gestion	El juego permite borrar una partida previamente empezada	-
6-rf-puntuacion	El juego permite visualizar la puntuación en un registro de partidas	-
7-rf-	La aplicación sólo muestra el listado de partidas terminadas	-

Requerimientos No Funcionales

TABLA 3 REQUERIMIENTOS NO FUNCIONALES

[nº-rnf-codigo]	Descripción	Validación	Métrica
1-rnf-red	La aplicación debe recuperarse de fallos por inconsistencias de la red	Si	Control de errores por fallo de la red
2-rnf-SO	La aplicación debe ser para el sistema operativo Android y el código debe ser reutilizable	No	-
3-rnf-Pantalla	El tamaño de pantalla del dispositivo móvil debe ser tenido en cuenta	Si	Prueba de adaptabilidad de la aplicación. Nivel de adaptabilidad: bajo o alto. Se adapta o no se adapta.
4-rnf-base de datos	La base de datos debe permitir el fácil añadido de datos	Si	Facilidad al añadir datos a la aplicación
5-rnf	La aplicación debe ser compatible con el formato estándar	No	

	Common Cartridge		
6-rnf	Se debe tener en cuenta la movilidad de los usuarios y que los eventos ocurren asíncronamente	-	-
7-rnf	El contenido y la gestión de usuarios se realiza dentro de la red de la institución, no en el dispositivo móvil	-	-

3.2 MOCK-UP

El mock-up para comprobar las funcionalidades de esta aplicación se realizó con la herramienta MIT App Inventor de Google. Para utilizar esta aplicación es necesario tener una cuenta de Gmail. Si ya se tiene esta cuenta, es necesario ingresar a la web de la aplicación y configurar las herramientas para utilizarla. En este caso se ha configurado MIT para utilizarla desde el ordenador aunque existe la opción de desplegar el mock-up en un dispositivo móvil.

3.2.1 SET UP

Para ejecutar el mock-up en el ordenador es necesario descargar y configurar un emulador de móvil para que trabaje con App Inventor.

Antes de comenzar con la parte de configuración es necesario tener en cuenta los requerimientos de sistema. La siguiente figura muestra la recomendación de la aplicación.

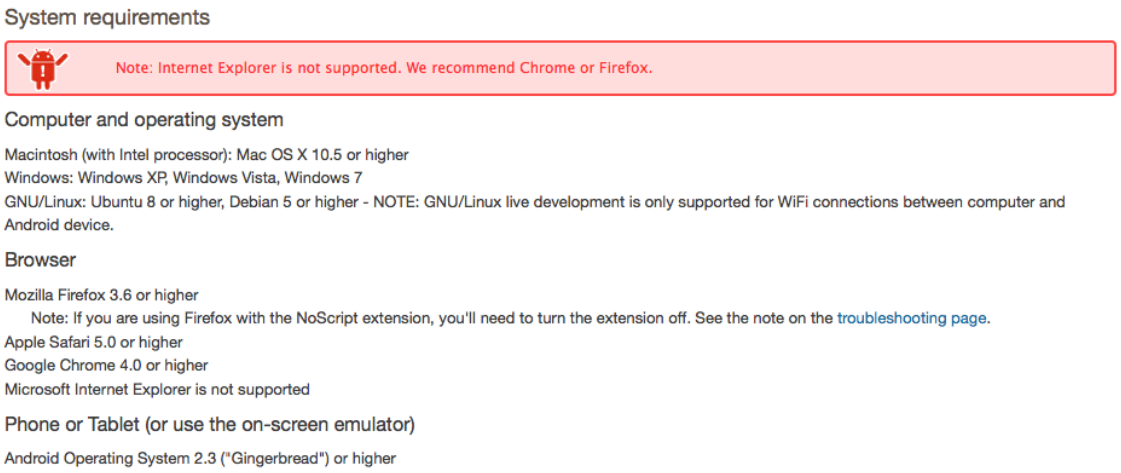


ILUSTRACIÓN 8 REQUERIMIENTOS DEL SISTEMA MIT[15]

Una vez que se tengan las herramientas requeridas según las recomendaciones de MIT hay que seguir los siguientes pasos:

1. Descargar el emulador[16]. En este caso se ha descargado la de Mac
2. Ejecutar aiInstaller. En Mac se ejecuta automáticamente[17]

Quizá sea necesario descargar e instalar o actualizar el software Java que tengamos instalado en el ordenador antes de proceder a ejecutar la aplicación MIT[18]

Después de haber configurado el emulador es necesario crear un proyecto y conectarlo al emulador. Esto se realiza yendo a la página principal de MIT y pulsando en el botón "CREATE!" en la parte derecha superior de la pantalla. Luego de pulsar el botón se abrirá una ventana que contiene la aplicación de diseño MIT.

Para crear un proyecto, se selecciona la opción "Start new project" dentro del menú desplegable: Projects.

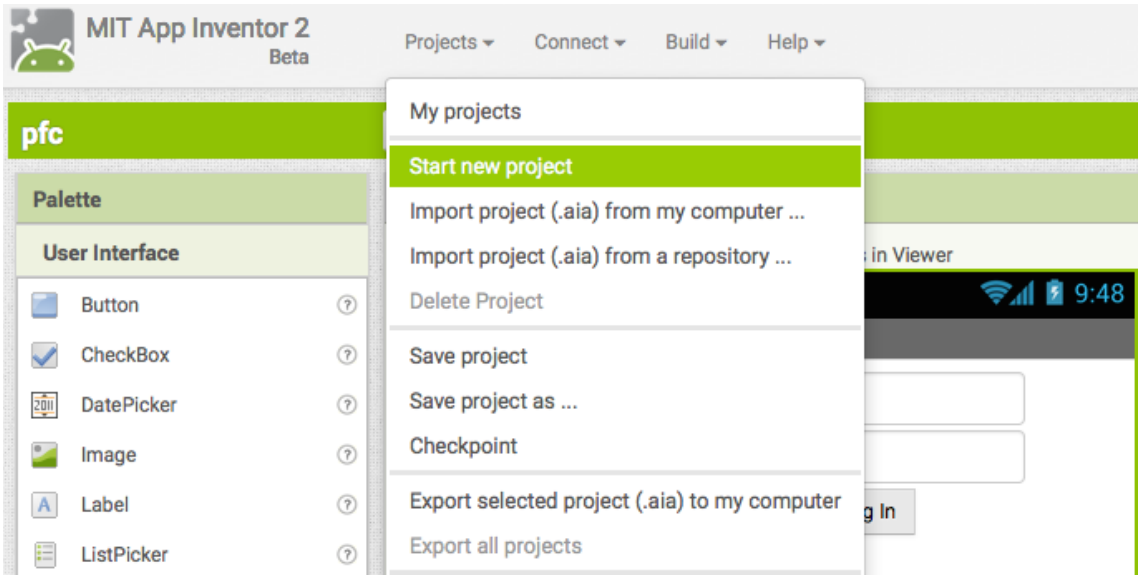


ILUSTRACIÓN 9 CREAR UN PROYECTO MIT

Después de haber creado el proyecto se procede a diseñarlo. En la siguiente figura se muestra la actividad de Log In de la aplicación que se quiere desarrollar.

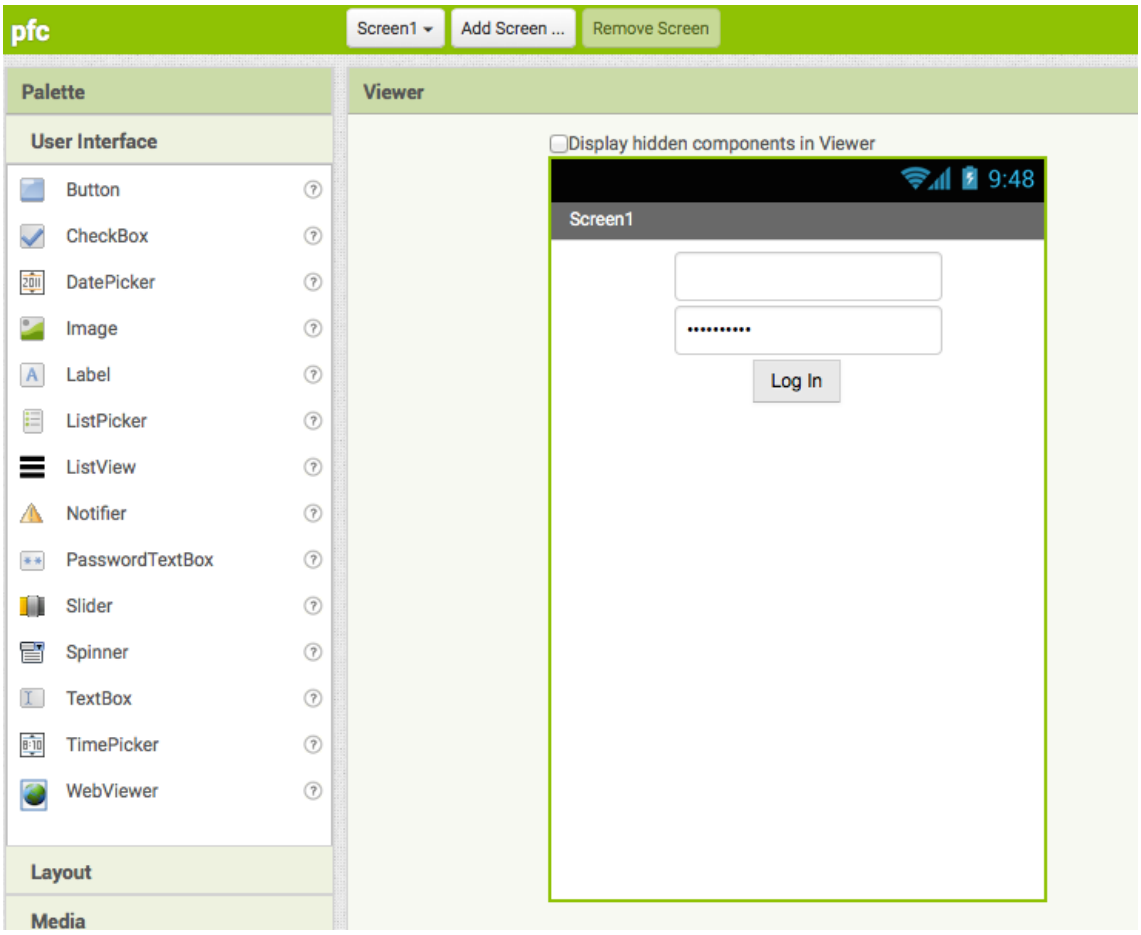


ILUSTRACIÓN 10 DISEÑO DE MOCK UP CON MIT

En este proyecto se crearon las actividades para jugar una partida, log in, ver sesiones, etc. Sin embargo no se muestran todas las actividades que se han diseñado porque MIT borra el contenido de un proyecto al cabo de un tiempo. Hay que tener en cuenta que esto es una aplicación gratis que no tiene recursos para el almacenamiento de documentos como si fuera un servicio tipo hosting de datos.

3.3 DISEÑO DE LA APLICACIÓN JUNTO CON LA HERRAMIENTA DE DISEÑO Y MODELO UTILIZADO

Una arquitectura es, según la terminología del estándar ISO/IEC/IEEE 42010:2011 [19], conceptos y propiedades de un sistema en su entorno que están incorporados en sus componentes, relaciones, y en los principios de su diseño y evolución. Esto quiere decir que la arquitectura de un sistema debe representar su estructura y comportamiento según los requerimientos y casos de uso definidos para su uso.

3.3.1 MODELO DEL ANÁLISIS

Para generar una arquitectura, existen patrones definidos para distintos tipos de sistema. El diseño de la arquitectura en este caso se ha basado en el patrón de arquitectura Modelo Vista Controlador (MVC) que es el que más se adapta a los requerimientos de interacción con el usuario, gestión del contenido y comunicación entre el contenido y la aplicación.

Este modelo, tal como indica el nombre, tiene tres componentes: (1) el Modelo, (2) la Vista y (3) el Controlador.

El componente (1) tiene como fin la representación de la información que maneja el sistema, el acceso a dicha información e incluye la lógica de negocio. El componente (2) se encarga de procesar eventos que provienen de la interacción del usuario con la Vista e invoca peticiones de información al Modelo. El último componente (3) se encarga de la interacción con el usuario y de presentar la información del Modelo de una forma apropiada.

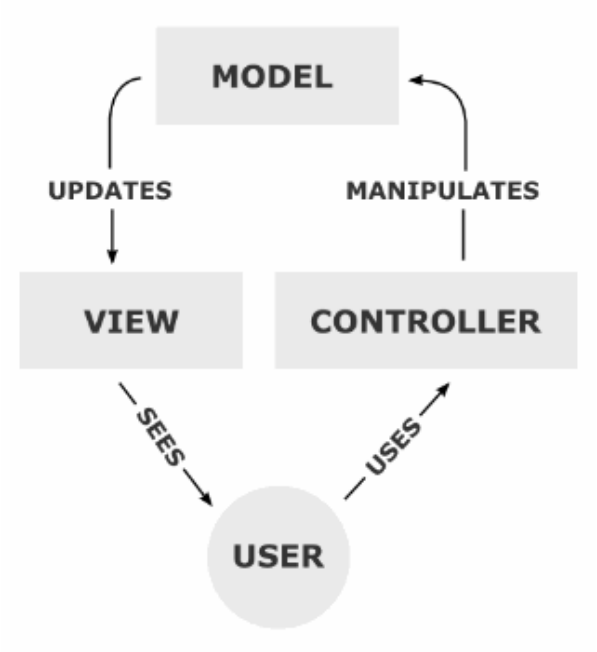


ILUSTRACIÓN 11 ARQUITECTURA DEL SISTEMA: MODELO VISTA CONTROLADOR

3.3.1.1 MVC Y SUBSISTEMAS

El MVC en este caso esta presentado en tres componentes: (1) ILLLabGUI que representa la Vista, (2) ILLLabData que representa el Modelo y (3)ILLLabFunctional que representa al Controlador.

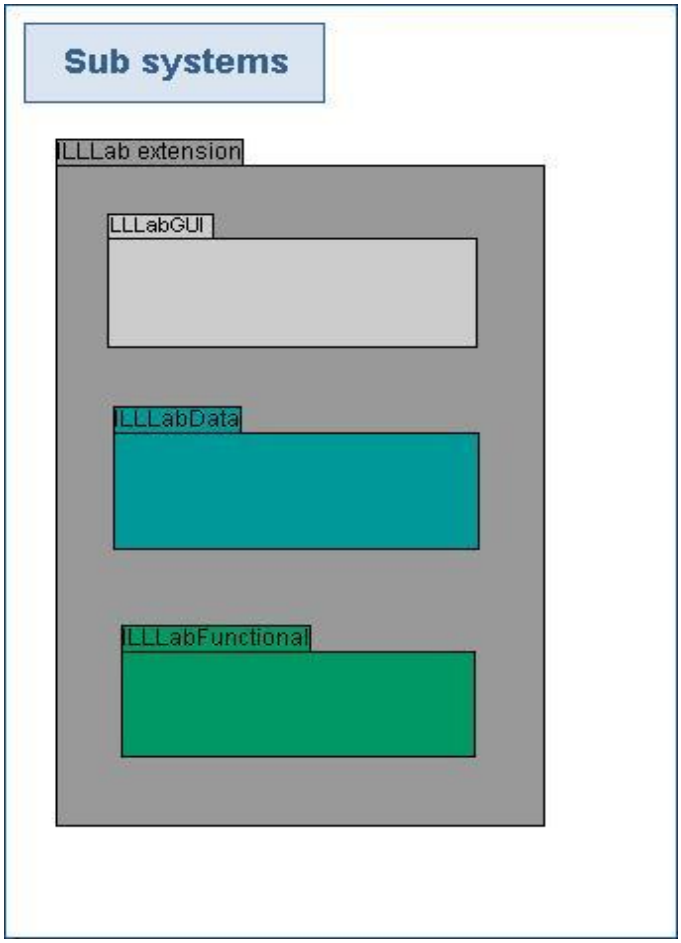
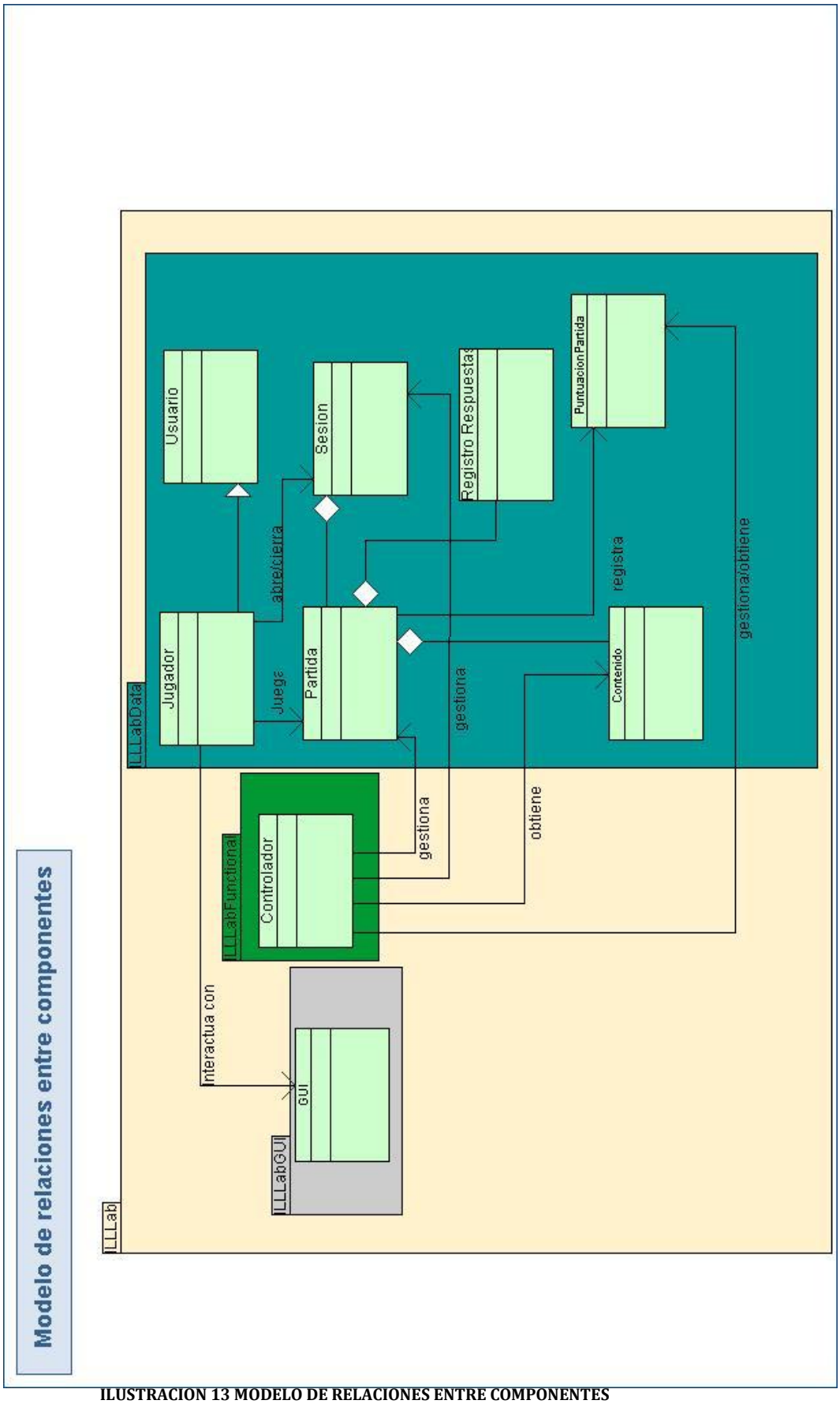


ILUSTRACIÓN 12 SUB SISTEMAS DE LA EXTENSION DE ILLLAB

Diagrama estructural

En vistas de entender un poco más sobre los elementos que interactúan en el sistema, se presenta un diagrama de objetos por subsistema y algunas de sus asociaciones. Esta vista muestra los objetos que se observan a primera vista sin indicar mayores interacciones o componentes que complementan o implementan funcionalidades extras para que el sistema esté operativo.



ILUSTRACION 13 MODELO DE RELACIONES ENTRE COMPONENTES

A partir de este modelo se pueden extraer otros derivados de la gestión de los objetos que se han mostrado anteriormente. El diagrama más inmediato es el de ILLLabData(Modelo) que además de los objetos que representan la información en sí también se incluyen los objetos que se encargan de gestionar la información y realizar la lógica de negocio (cálculos, cambios de estado en objetos, etc.)

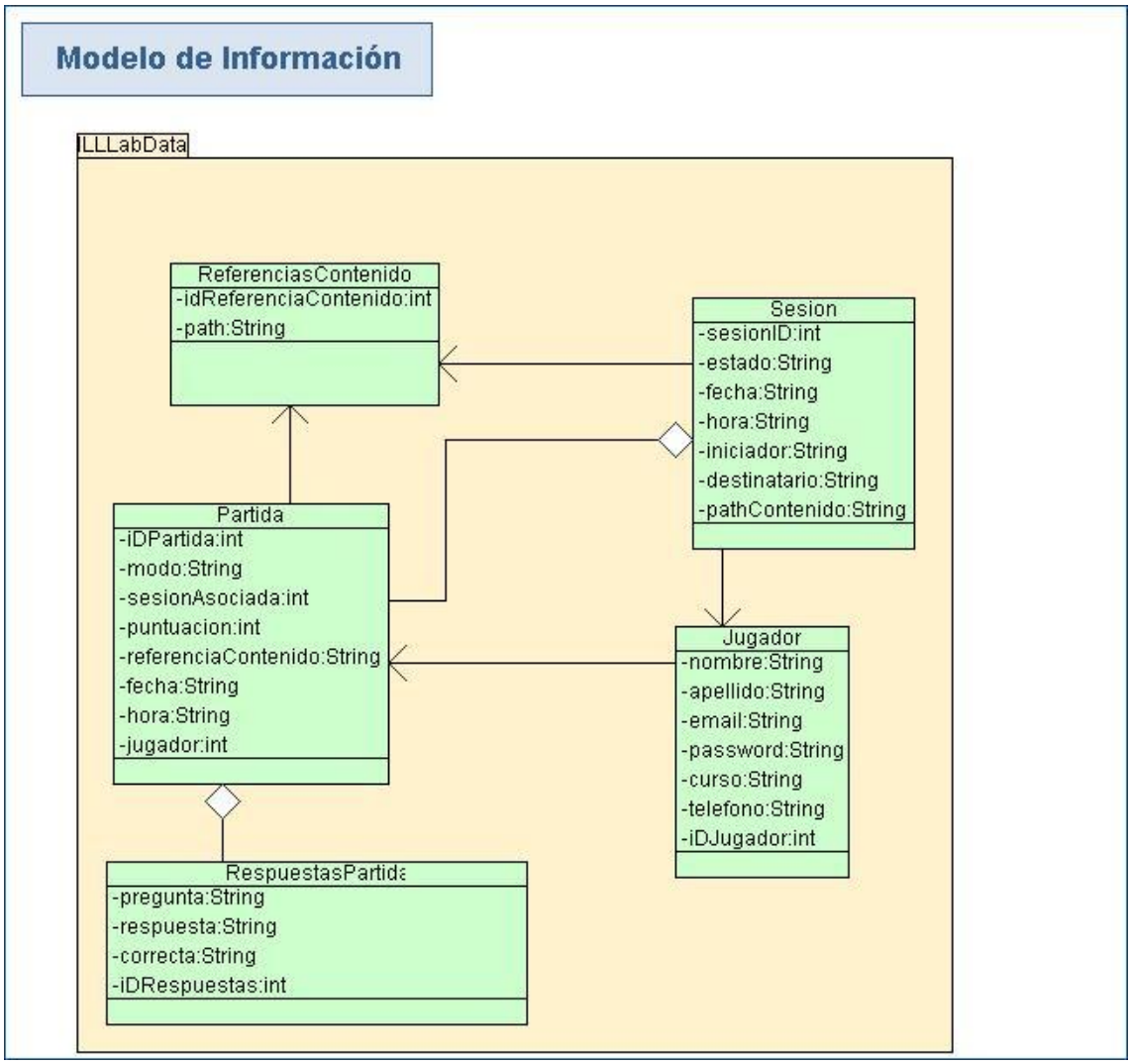


ILUSTRACIÓN 14 MODELO DE INFORMACION

El siguiente diagrama muestra la relación entre los componentes funcionales de la arquitectura con los métodos que se definen como resultado del modelo del diseño que se presenta en la siguiente sección. En esta sección sólo es necesario prestar atención a los objetos con los que interactúa el Controlador (ILLLab Functional) ya que es el paso para explicar las relaciones entre objetos en el modelo del diseño en el cual se especifica la implementación de éstas relaciones.

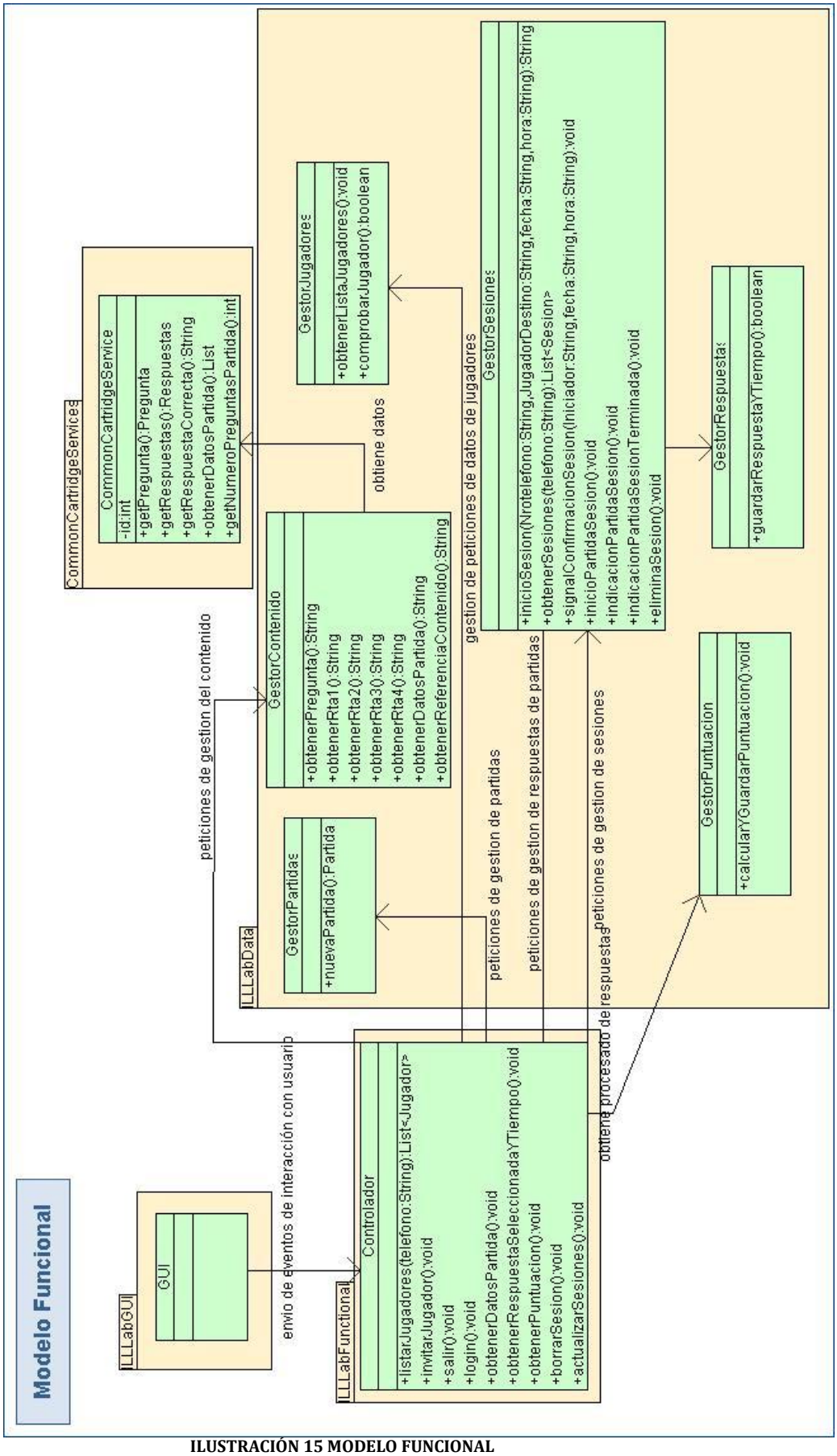


ILUSTRACIÓN 15 MODELO FUNCIONAL

Una vez se tiene clara la estructura de los componentes que componen la aplicación se procede a analizar el comportamiento en rasgos generales. La aplicación tiene principalmente dos estados, uno cuando el jugador no se ha autenticado y la otra es cuando el jugador ya ha accedido y quiere gestionar partidas o sesiones (visualizar, crear, etc.) o jugar.

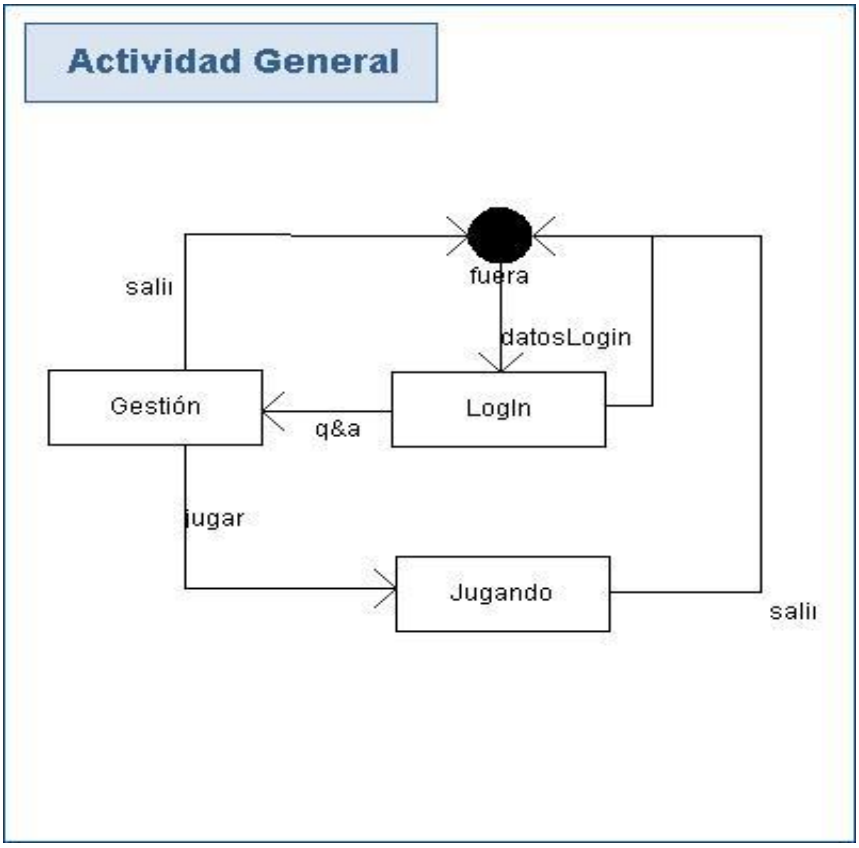


ILUSTRACIÓN 16 DIAGRAMA DE ACTIVIDAD GENERAL DE LA APLICACION

Este es el límite entre el modelo del análisis y el modelo del diseño. De aquí en más se debe representar el comportamiento de la aplicación mediante la implementación de los casos de uso que se presentan en la siguiente sección.

3.3.2 MODELO DEL DISEÑO

El modelo del diseño se basa en la implementación de los casos de uso definidos en la sección 2.1. Esto incluye diagramas de actividad de para especificar la secuencia lógica de eventos que realiza la aplicación para funcionar y diagramas de secuencia para especificar cómo se llevan a cabo los casos de uso desde el punto de vista de la comunicación entre objetos de distintos componentes de la arquitectura. Esto, al final, define los métodos que ofrece cada objeto como interfaz para el resto.

3.3.2.1 DIAGRAMAS DE ACTIVIDAD

Invitar Jugador

El siguiente diagrama indica la secuencia de eventos para que Jugador A invite a Jugador B

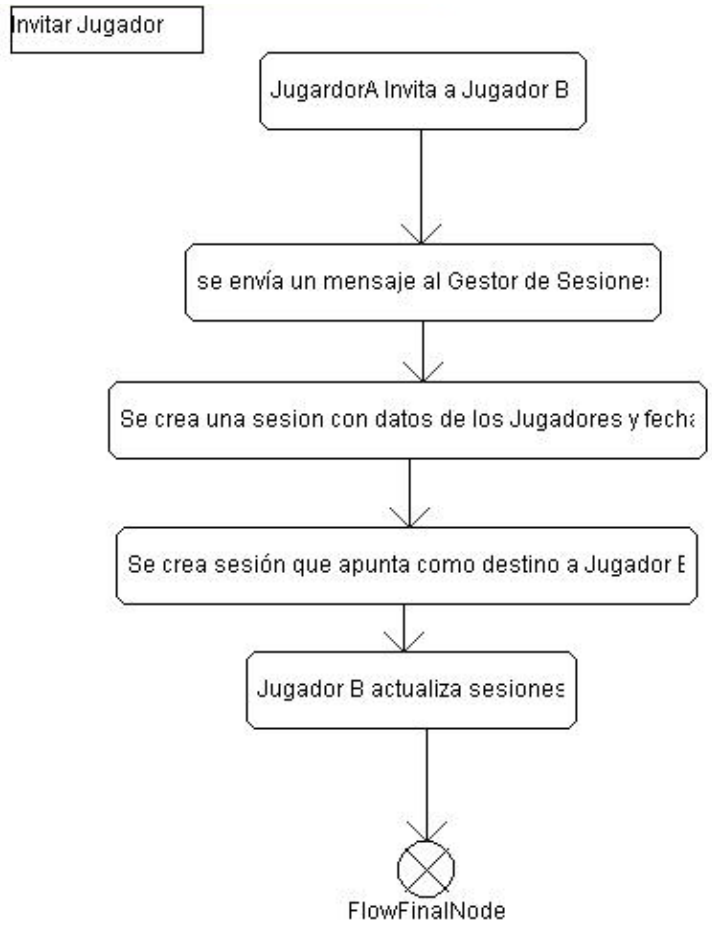


ILUSTRACIÓN 17 DIAGRAMA DE ACTIVIDAD: INVITAR JUGADOR

Confirmación/Rechazo Invitación

Al recibir una invitación, el Jugador B la observa en la lista de sesiones activas y procede a rechazarla o a confirmarla para luego jugar con Jugador A.

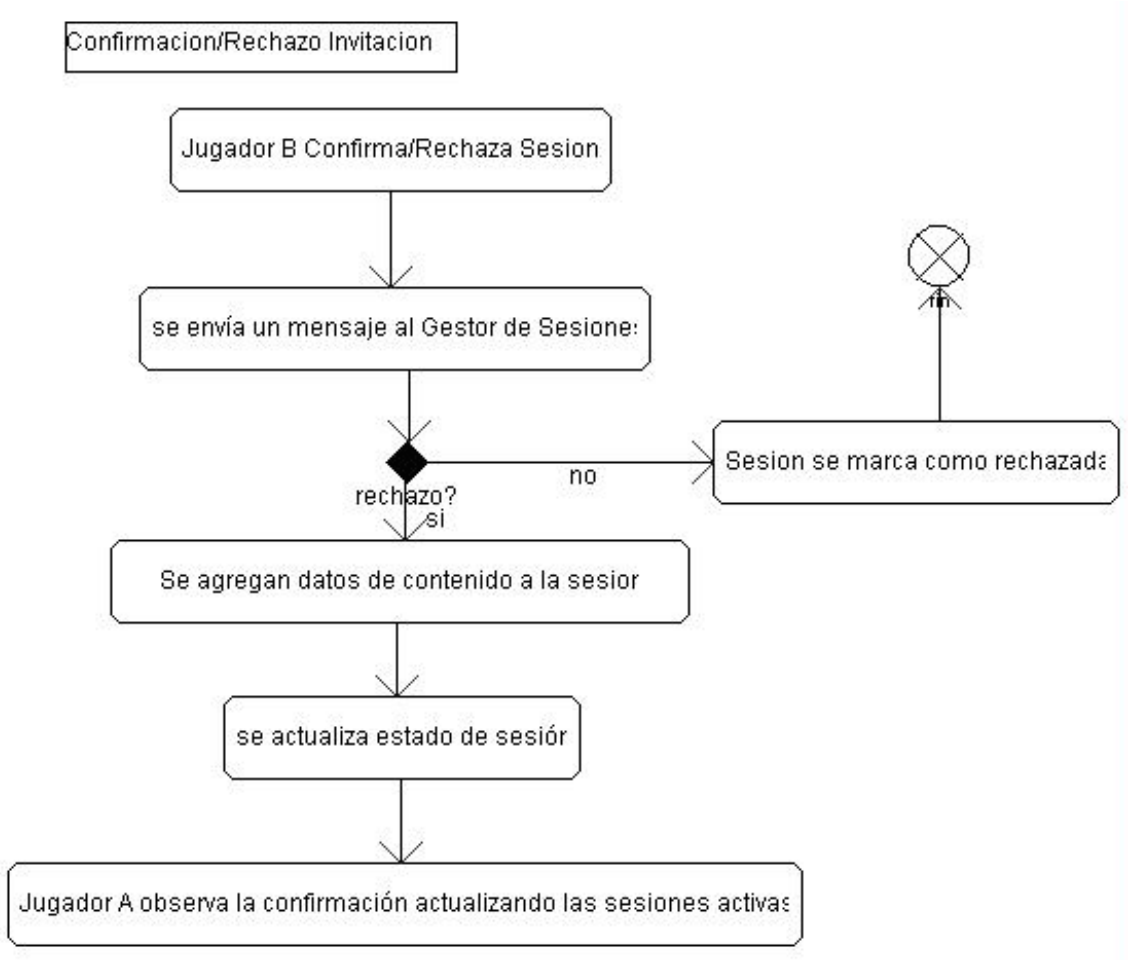


ILUSTRACIÓN 18 DIAGRAMA DE ACTIVIDAD: CONFIRMACION/RECHAZO INVITACION

Inicio Partida Nueva + Jugar Partida

El siguiente diagrama muestra los eventos que tienen lugar desde la creación de una sesión con Jugador B hasta que el Jugador A juega una partida e indica a Jugador B que ha jugado.

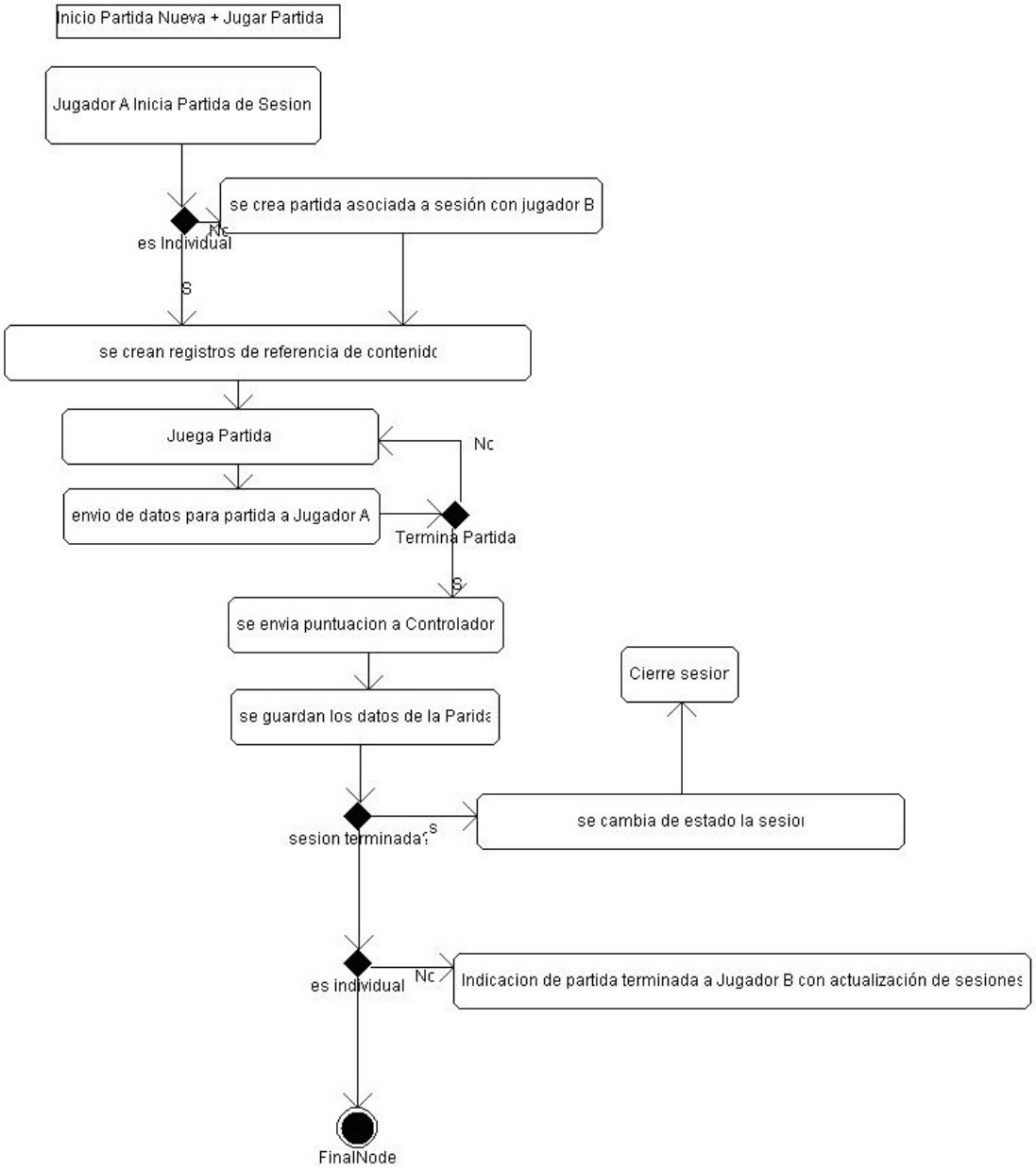


ILUSTRACIÓN 19 DIAGRAMA DE ACTIVIDAD: INICIAR PARTIDA Y JUGAR PARTIDA

Gestionar Sesiones

El usuario entra en la sección de visualización de sesiones. A continuación se muestra una pantalla con dos opciones: mostrar sesiones activas ó mostrar sesiones terminadas. Las acciones que se pueden realizar sobre las sesiones depende de su estado:

- Acciones sobre sesiones enviadas: Ver info, borrar
- Acciones sobre sesiones recibidas: Ver info, confirmar, rechazar
- Acciones sobre sesiones confirmadas: Ver info, jugar partida
- Acciones sobre sesiones terminadas: Ver info

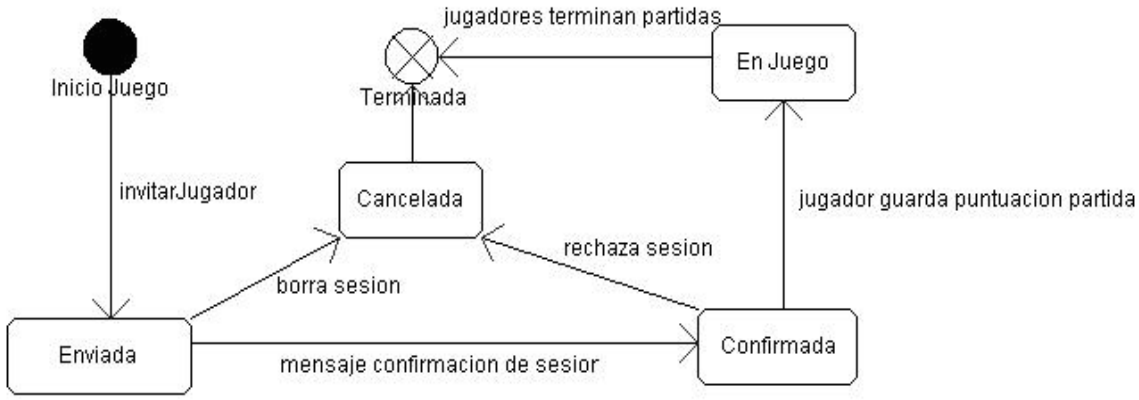


ILUSTRACIÓN 20 DIAGRAMA DE ESTADOS DE UNA SESION

Las sesiones activas se muestran juntas, que son las que se envían como invitación, las que se reciben, las confirmadas y las que están en juego. Cuando el jugador selecciona una se abre la pantalla de visualización y se muestran los datos y las acciones dependiendo de cuál se selecciona.

3.3.2.2 DIAGRAMAS DE SECUENCIA Y ACTIVIDAD DE LOS CASOS DE USO

Los diagramas de secuencia implementan lo que se ha definido en los diagramas de actividad anteriores pero entrando en los detalles de las funciones que deben cumplir cada uno de los componentes. En caso de que una función no se pueda definir con éstos diagramas se agregan diagramas de actividad que indica el procesado que realiza un componente internamente.

Login

El acceso a la aplicación se realiza mediante un portal de acceso con los datos de registro del usuario. En este caso es el correo y la contraseña que se utiliza para tener acceso a los servicios de la Universidad Politecnica de Madrid o la institución desde la que se esté gestionando la aplicación.

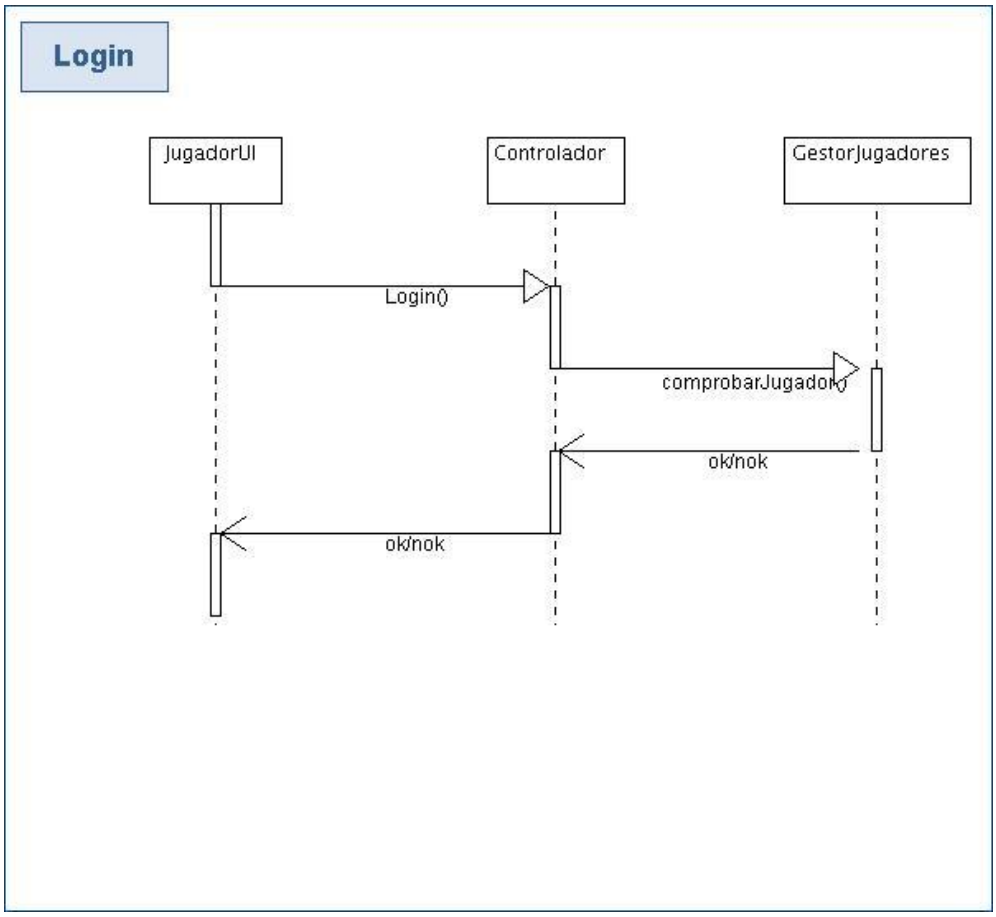


ILUSTRACIÓN 21 SECUENCIA: LOGIN

TABLA 4 METODOMÉTODO LOGIN

login(). Servicio de objeto Controlador	
Parámetros de entrada	• Correo de Jugador

	• Contraseña de jugador • Numero de teléfono del móvil del jugador
Parámetros de salida	• Confirmación de acceso al jugador
Consideraciones de implementación	El número de móvil no es introducido por el Jugador, se obtiene con funciones de las librerías Android. Debe haber una tabla en la base de datos del registro de jugadores con sus respectivos correos electrónicos, contraseñas, nombre y número de móvil.
Operaciones afectadas	ComprobarJugador()

Invitar Jugador

Una vez el jugador ha accedido a la aplicación, para comenzar una sesión con otro usuario es necesario realizar una invitación la cual debe ser aceptada por el jugador invitado.

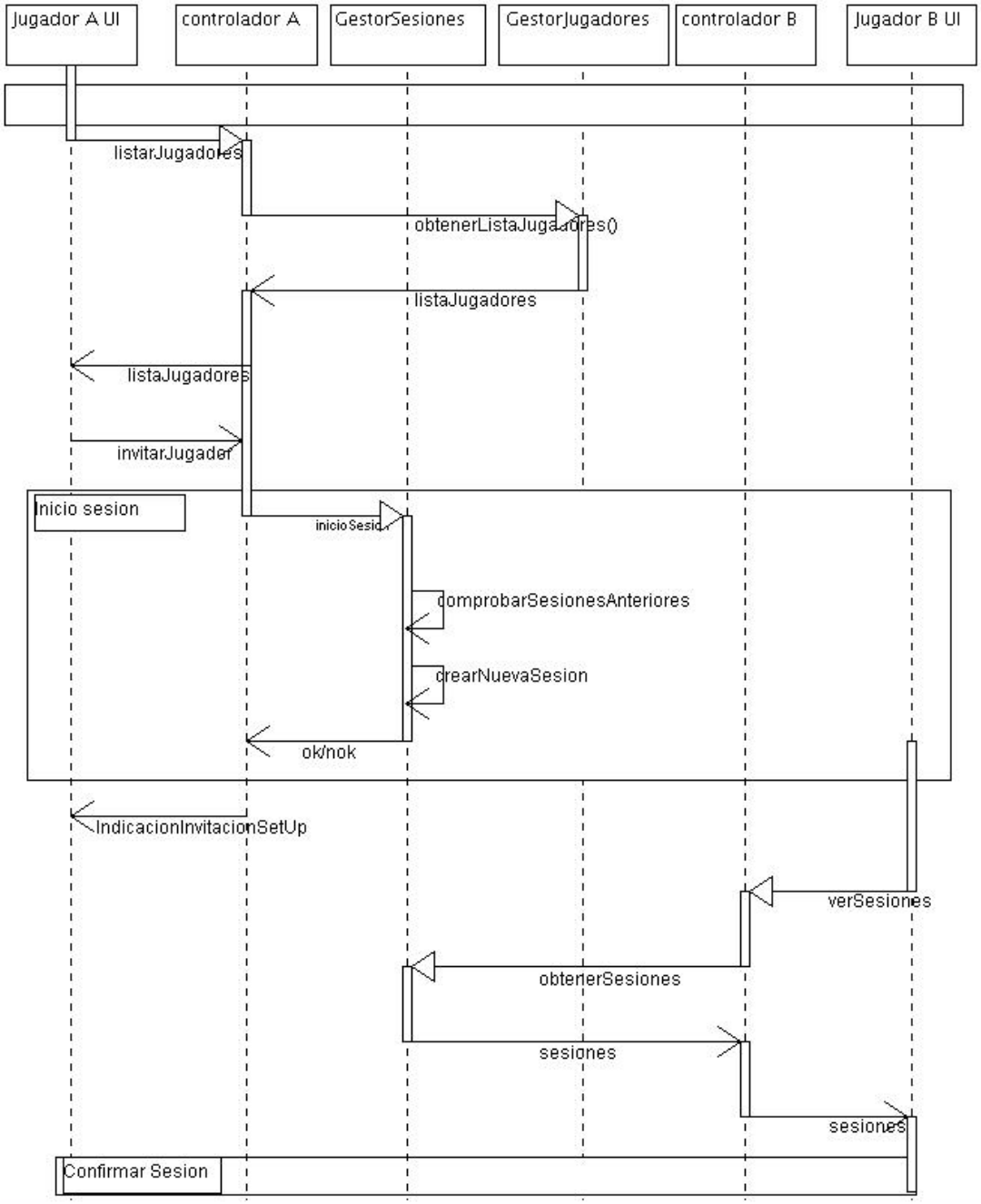


ILUSTRACIÓN 22 SECUENCIA: INVITAR JUGADOR

TABLA 5 MÉTODO LISTAR JUGADORES

listarJugadores(). Servicio de objeto Controlador en interfaz con UI	
Parámetros de entrada	• Número de teléfono de jugador
Parámetros de salida	• Lista de jugadores a mostrar
Consideraciones de implementación	El objeto Jugador debe tener un campo de número de teléfono que permita identificarlo y así poder listar todos los jugadores disponibles menos al jugador mismo. El listado de jugadores deben ser los que están en su misma clase
Operaciones afectadas	obtenerListJugadores(). Obtiene los mismos parámetros de entrada y crea una lista de sesiones

TABLA 6 MÉTODO INVITAR JUGADOR

invitarJugador(). Servicio de objeto Controlador en interfaz con UI	
Parámetros de entrada	• Número de teléfono de jugador iniciador • Nombre jugador destino • Fecha (dd/mm/yyyy) • Hora (hh:mm)
Parámetros de salida	• Confirmación de invitación enviada o de sesión existente
Consideraciones de implementación	--
Operaciones afectadas	--

TABLA 7 MÉTODO INICIO SESION

inicioSesion(). Servicio de objeto SesionBroker en interfaz con Controlador	
Parámetros de entrada	• Nro Telefono Jugador Iniciador • Nombre Jugador Destino • Fecha (dd/mm/yyyy) • Hora (hh:mm)
Parámetros de salida	• Confirmación de creación de sesión o de sesión existente
Consideraciones de implementación	Es necesario comunicar mediante un diálogo si ya existe una sesión
Operaciones afectadas	ComprobarSesionesAnteriores() crearNuevaSesion()

TABLA 8 MÉTODO COMPROBAR SESIONES ANTERIORES

ComprobarSesionesAnteriores(). Servicio de objeto SesionBroker	
Parámetros de entrada	• Nro Telefono Jugador Iniciador • Nombre Jugador Destino
Parámetros de salida	• Confirmación de sesión existente o no existente
Consideraciones de implementación	Comprueba si existen sesiones abiertas del jugador iniciador con el jugador destino
Operaciones afectadas	--

TABLA 9 MÉTODO CREAR NUEVA SESION

crearNuevaSesion(). Servicio de objeto SesionBroker

Parámetros de entrada	• Nro Telefono Jugador Iniciador • Nombre Jugador Destino • Fecha (dd/mm/yyyy) • Hora (hh:mm)
Parámetros de salida	• Confirmación de creación de sesión o de sesión existente
Consideraciones de implementación	Sólo se crea si no existe sesión previa con el jugador seleccionado
Operaciones afectadas	--

Ver Sesiones/Actualizar sesiones

Actualizar las sesiones es parte de la acción de visualizar sesiones y en los casos en que se tengan que actualizar las listas por cambio de estado en las sesiones

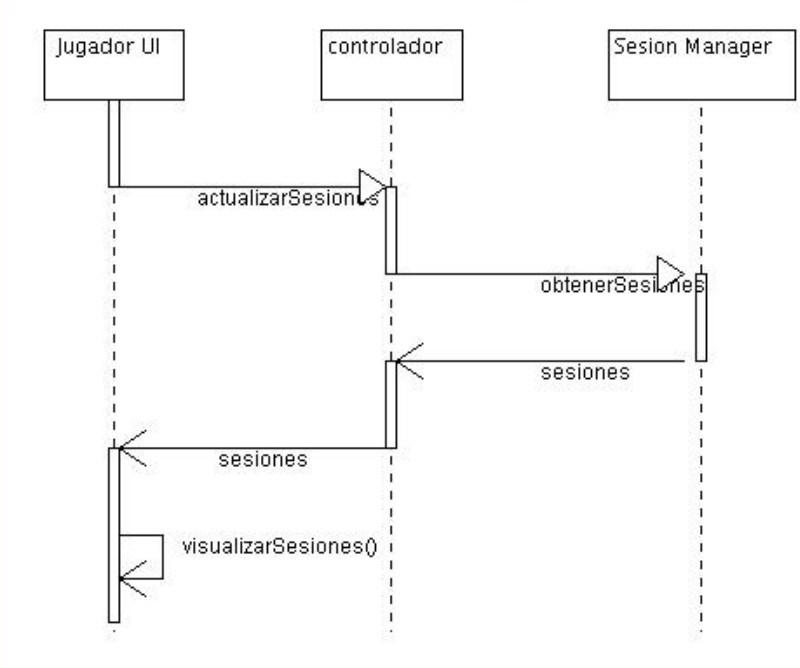


ILUSTRACIÓN 23 SECUENCIA: VER SESIONES Y ACTUALIZAR SESIONES

TABLA 10 MÉTODO ACTUALIZAR SESIONES

actualizarSesiones(). Servicio de objeto Controlador en interfaz con UI	
Parámetros de entrada	• Número de teléfono de jugador • Tipos de sesiones a listar: Activas/Terminadas
Parámetros de salida	• Lista de sesiones a mostrar
Consideraciones de implementación	El objeto sesión debe tener un campo de estado que permita identificar su estado actual y así poder separar las sesiones activas de las terminadas.
Operaciones afectadas	obtenerSesiones(). Obtiene los mismos parámetros de entrada y crea una lista de sesiones

Borra/rechaza sesión

Esta funcionalidad no se implementa debido a que no se puede borrar una sesión por mantener un registro las sesiones creadas. Inicialmente se había pensado Sólo se puede borrar una sesión que no haya pasado al estado Confirmado. En caso de que el iniciador de la sesión quiera borrarla entonces la partida debe estar en estado Enviada. En el caso del destinatario de la sesión, sólo se rechazará la sesión.

Las sesiones terminadas quedan registradas para la evaluación y pueden ser borradas manualmente desde una herramienta de gestión de datos.

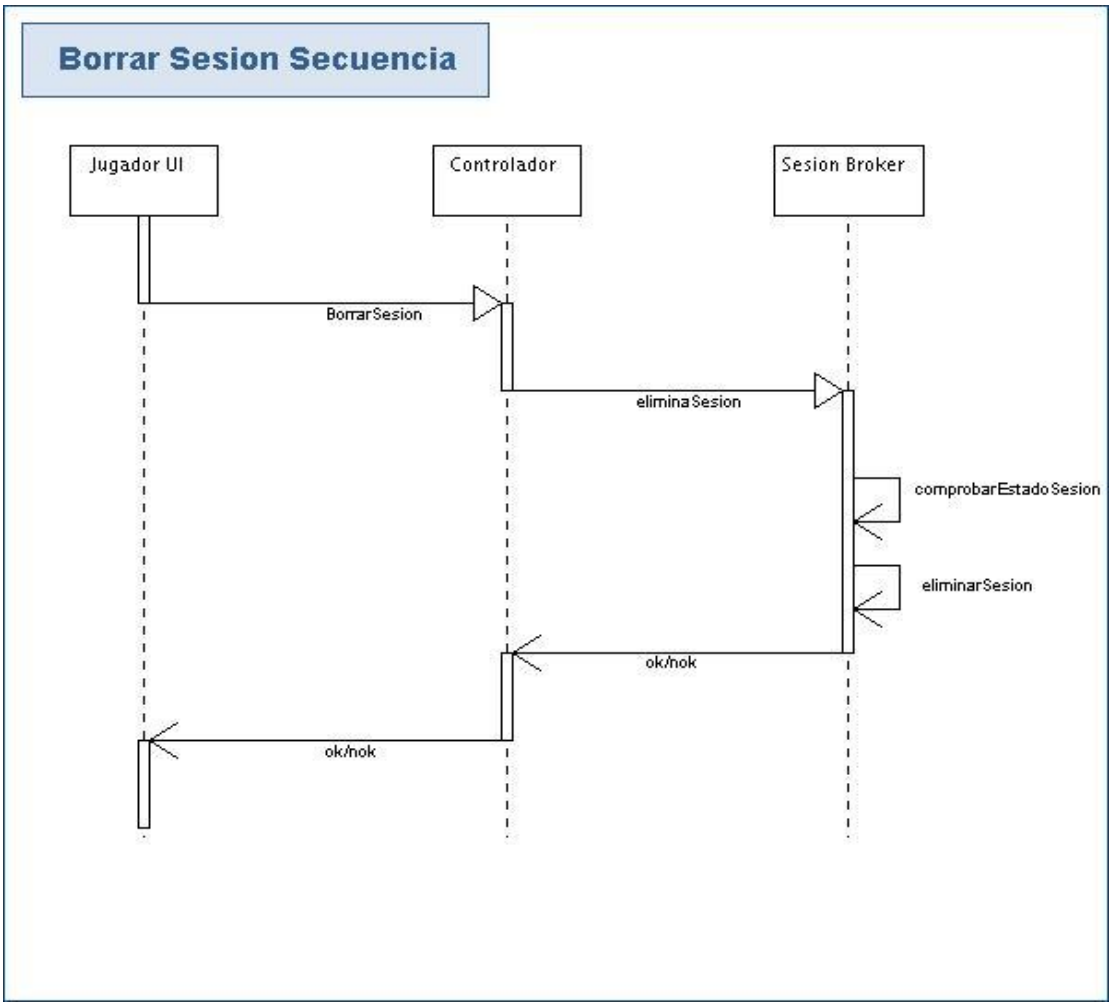


ILUSTRACIÓN 24 SECUENCIA: BORRAR SESION

TABLA 11 SECUENCIA: BORRAR SESION

borrarSesion(). Servicio de objeto Controlador en interfaz con UI	

Parámetros de entrada	• Datos de sesión:
	  ○ Nombre Iniciador
	  ○ Fecha
	  ○ Hora
	  ○ ID

Parámetros de salida	• Confirmación de borrado de sesión. En caso de que no se pueda borrar es porque no existe o porque se ha confirmado

Consideraciones de implementación	El objeto sesión debe tener un campo de estado que permita identificar su estado actual y así poder separar las sesiones activas de las terminadas.
Operaciones afectadas	eliminaSesion(). Obtiene los mismos parámetros de entrada y crea una lista de sesiones

Confirmación Invitación

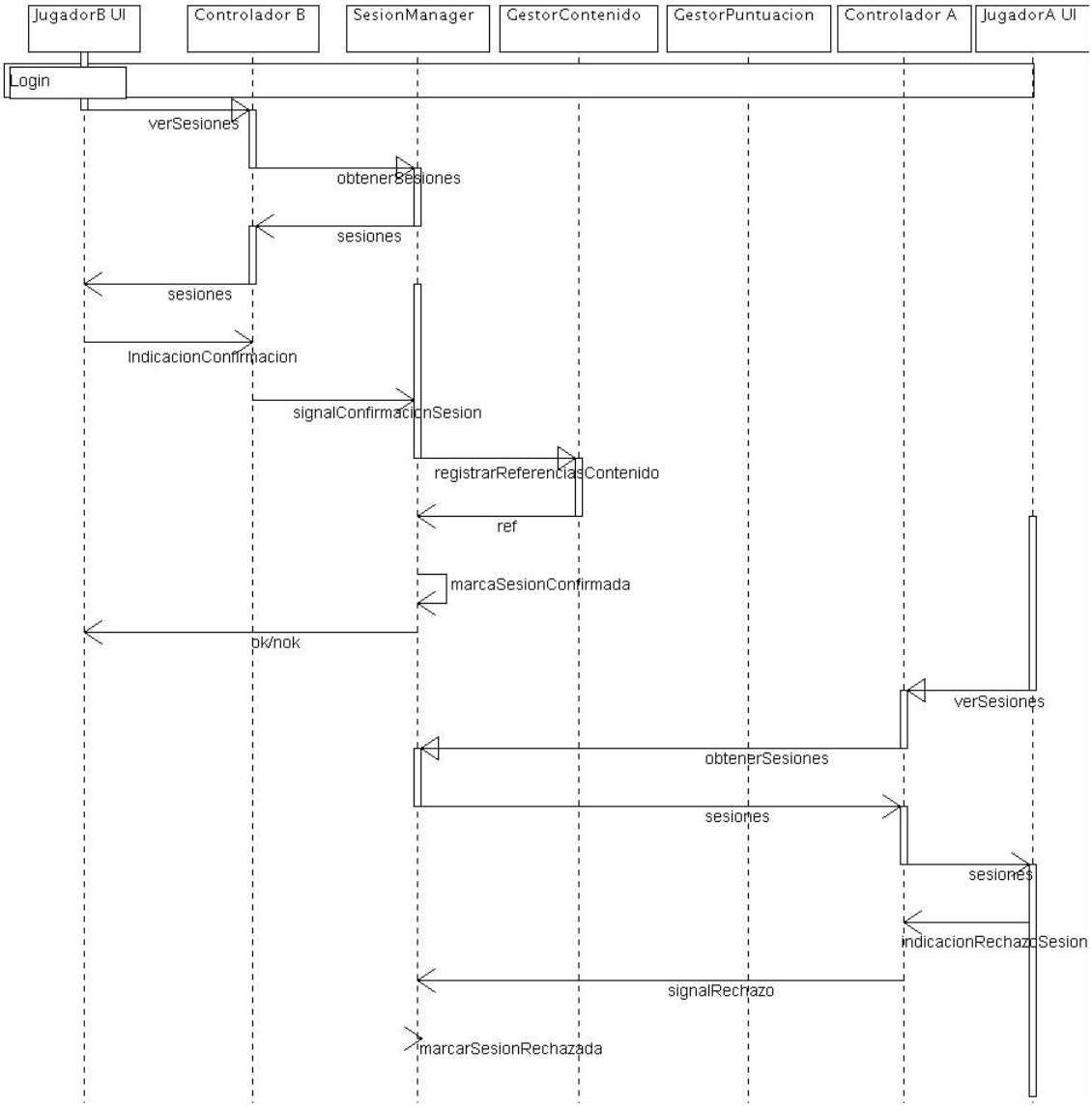


ILUSTRACIÓN 25 SECUENCIA: CONFIRMAR INVITACION

TABLA 12 MÉTODO: INDICACION CONFIRMACION

indicacionConfirmacion(). Servicio de objeto Controlador en interfaz con UI	
Parámetros de entrada	• Teléfono del jugador que confirma • Sesión asociada: ID Sesión
Parámetros de salida	• Confirmación de cambio de estado de sesión
Consideraciones de implementación	--
Operaciones afectadas	obtenerReferenciaContenido()

TABLA 13 MÉTODO SIGNAL CONFIRMACION SESION

signalConfirmacionSesion(). Servicio de objeto en SesionManager interfaz con Controlador	
Parámetros de entrada	• Teléfono del jugador que confirma • Sesión asociada: ID Sesión
Parámetros de salida	• Confirmación de cambio de estado de sesión
Consideraciones de implementación	--
Operaciones afectadas	obtenerReferenciaContenido() crearRegistroPuntuaciones() marcarSesionConfirmada()

TABLA 14 MÉTODO: INDICACION RECHAZO SESION

indicacionRechazoSesion(). Servicio de objeto en Controlador interfaz con UI	
Parámetros de entrada	• Teléfono del jugador que rechaza

	• Sesión asociada: ID Sesión
Parámetros de salida	• Confirmación de cambio de estado de sesión
Consideraciones de implementación	--
Operaciones afectadas	obtenerReferenciaContenido() crearRegistroPuntuaciones() marcarSesionConfirmada()

Jugar Partida

A continuación se define el proceso que se realiza al jugar una partida. Cada partida es una actividad dentro de la secuencia de partidas que corresponden al juego. El usuario puede jugar hasta 10 partidas (Ilustración 26).

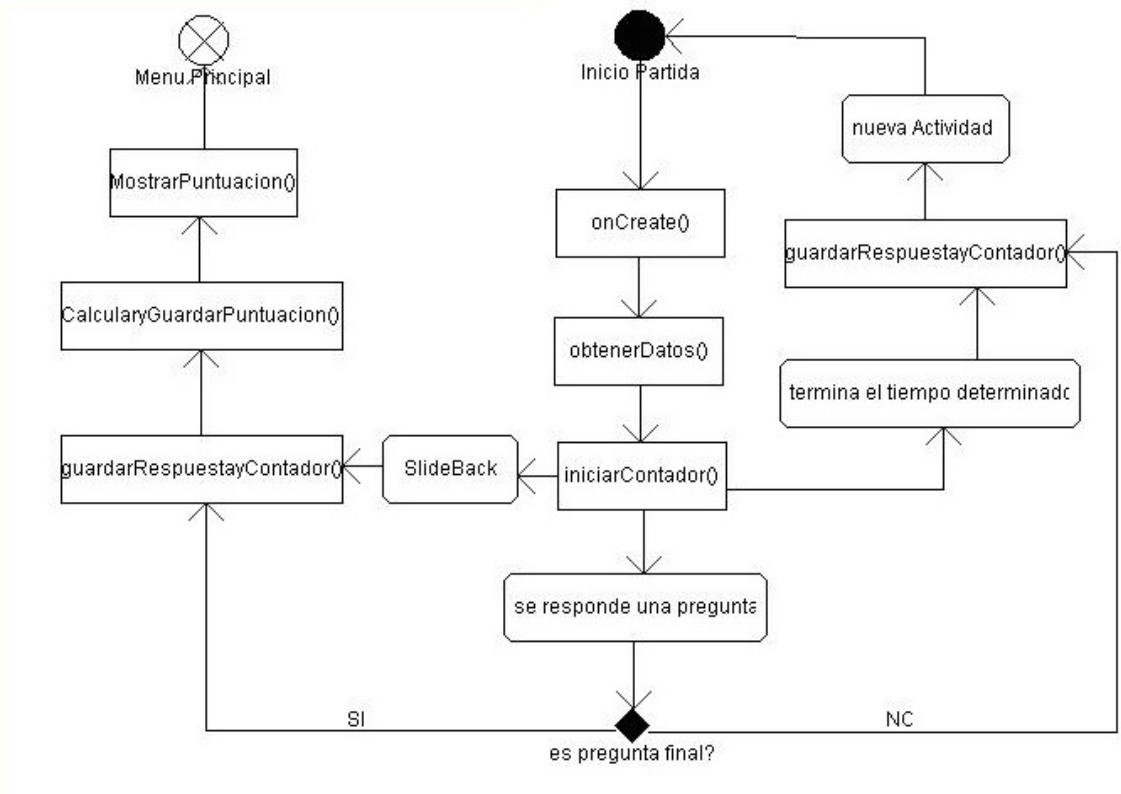


ILUSTRACIÓN 26 DIAGRAMA DE ACTIVIDAD: JUGAR UNA PARTIDA

Este caso de uso presenta el caso de una partida en modo individual y una extensión que es el caso en que se invita a otro jugador.

El jugador selecciona el modo individual y a continuación se crean registros para guardar los datos de la partida. Los datos de la partida son:

- Teléfono del jugador que inicia la partida
- Fecha
- Hora
- Tipo de partida: Individual/Doble
- Respuestas de la partida y tiempo
- Sesión asociada en caso de que sea doble

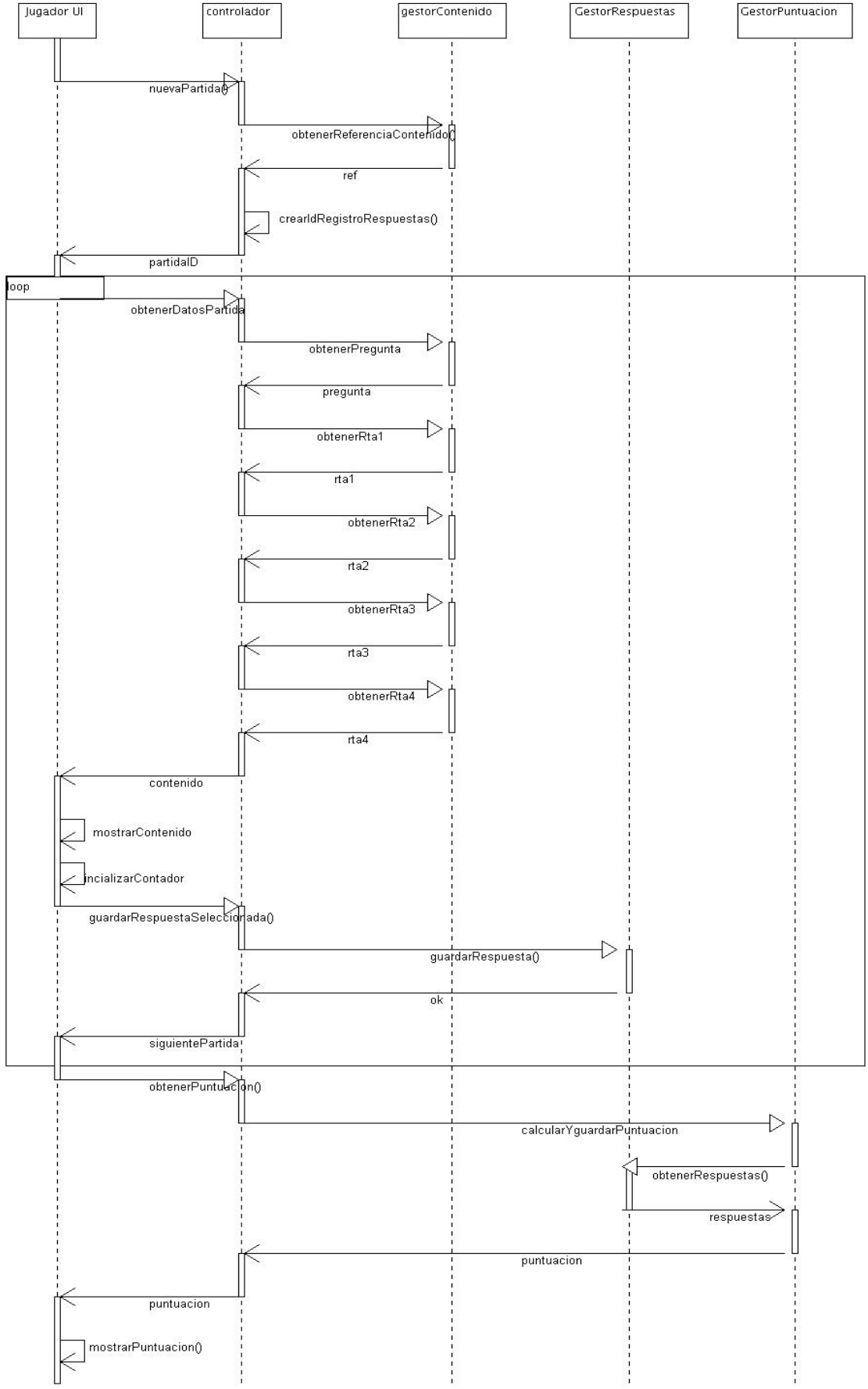


ILUSTRACIÓN 27 SECUENCIA: JUGAR PARTIDA INDIVIDUAL

TABLA 15 MÉTODO NUEVA PARTIDA

nuevaPartida(). Servicio de objeto Controlador en interfaz con UI	
Parámetros de entrada	• Teléfono del jugador que inicia la partida • Fecha • Hora • Tipo de partida: Individual/Doble • Sesión asociada: ID Sesión (en caso de partida individual=null)
Parámetros de salida	• Pregunta y respuestas de la partida
Consideraciones de implementación	El objeto Partida debe tener los datos indicados en los parámetros de entrada
Operaciones afectadas	--

TABLA 16 MÉTODO OBTENER DATOS PARTIDA

obtenerDatosPartida(). Servicio de objeto Controlador en interfaz con UI	
Parámetros de entrada	• Teléfono del jugador que inicia la partida • Fecha • Hora • Tipo de partida: Individual/Doble • Sesión asociada: ID Sesión
Parámetros de salida	• Pregunta y respuestas de la partida
Consideraciones de implementación	El objeto Partida debe tener los datos indicados en los parámetros de entrada
Operaciones afectadas	--

Jugar Partida Sesión

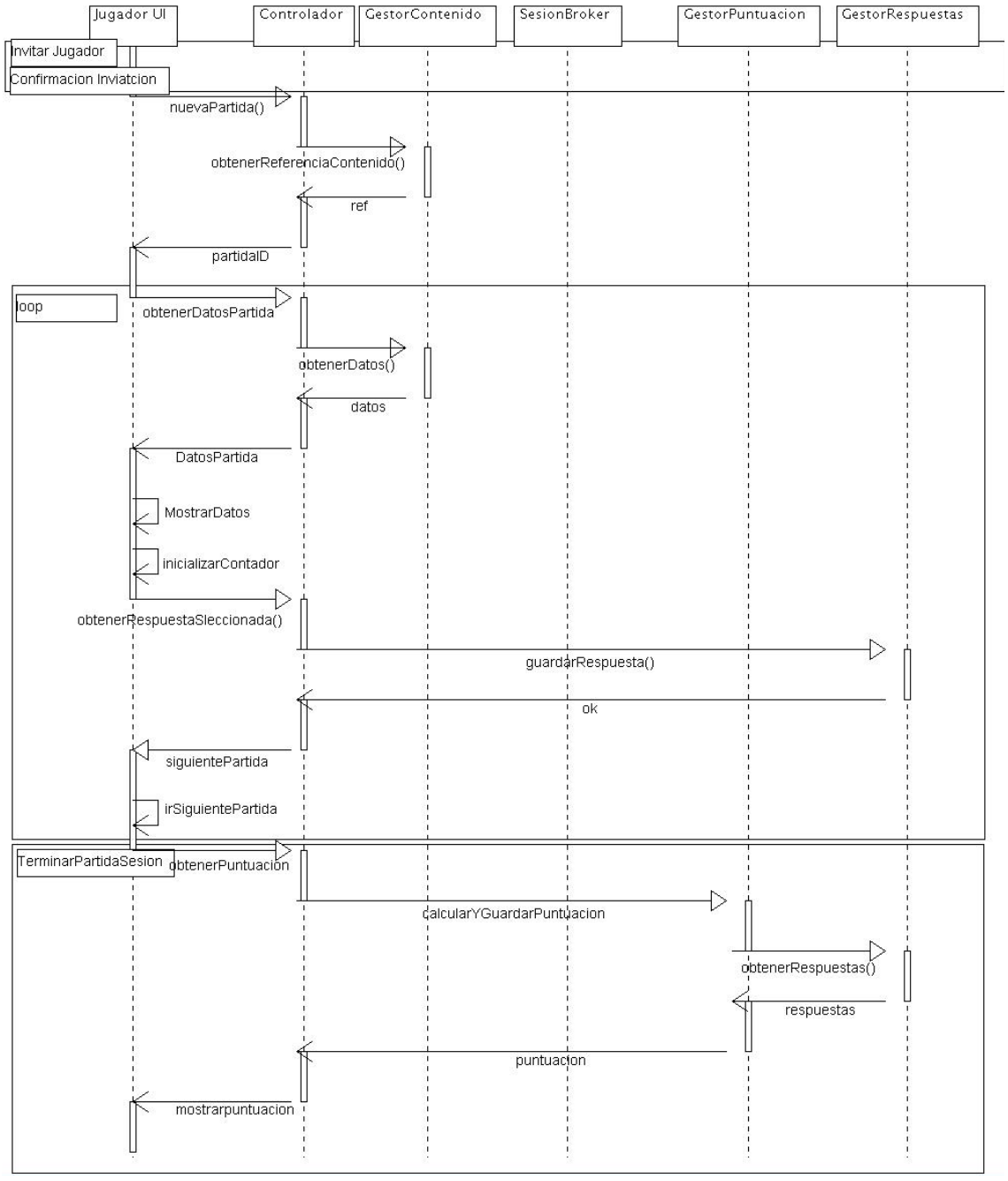


ILUSTRACIÓN 28 SECUENCIA: JUGAR PARTIDA EN SESION

TABLA 17 MÉTODO INICIO PARTIDA SESION

inicioPartidaSesion(). Servicio de objeto SesionBroker en interfaz con Controlador	
Parámetros de entrada	• Teléfono del jugador que inicia la partida • Fecha • Hora • Sesión asociada: ID Sesión
Parámetros de salida	• Confirmación de creación de registro de puntuación y respuestas de la partida
Consideraciones de implementación	--
Operaciones afectadas	obtenerReferenciaContenido()

TABLA 18 MÉTODO OBTENER REFERENCIA CONTENIDO

obtenerReferenciaContenido(). Servicio de objeto SesionBroker	
Parámetros de entrada	• ID Sesión
Parámetros de salida	• ID Referencia al Contenido
Consideraciones de implementación	--
Operaciones afectadas	--

TABLA 19 MÉTODO OBTENER DATOS

obtenerDatos(). Servicio de objeto Controlador. Esta funcionalidad resume el caso de uso Inicio Partida Nueva donde se obtienen los datos de una partida	
Parámetros de entrada	• ID Referencia Contenido
Parámetros de salida	• Datos de la partida
Consideraciones de implementación	--

Operaciones afectadas	--

TABLA 20 MÉTODO OBTENER RESPUESTA SELECCIONADA

obtenerRespuestaSeleccionada(). Servicio de objeto Controlador en interfaz con UI	
Parámetros de entrada	• Respuesta Seleccionada • Tiempo de cronómetro • ID Partida
Parámetros de salida	• Confirmación de guardado de datos
Consideraciones de implementación	Si no se pueden guardar los datos por algún problema, se sale de la partida sin guardar los datos y se indica al jugador que puede reiniciar la partida porque no se han guardado los datos Se reinicia el objeto partida y sus datos asociados
Operaciones afectadas	guardarRespuesta(). Realiza la operación sobre el registro de datos

TABLA 21 MÉTODO GUARDAR RESPUESTA

guardarRespuesta(). Servicio de objeto Controlador. Esta funcionalidad resume el caso de uso Inicio Partida Nueva donde se obtienen los datos de una partida	
Parámetros de entrada	• Respuesta Seleccionada • Tiempo de cronómetro • ID Partida
Parámetros de salida	• Confirmación de guardado de datos o notificación de error
Consideraciones de implementación	--

Operaciones afectadas	--

TABLA 22 MÉTODO OBTENER PUNTUACION

obtenerPuntuacion(). Servicio de objeto Controlador en interfaz con UI	
Parámetros de entrada	• Respuesta Seleccionada • Tiempo de cronómetro • ID Partida
Parámetros de salida	• Confirmación de guardado de datos o notificación de error
Consideraciones de implementación	Si no se pueden guardar los datos por algún problema, se sale de la partida sin guardar los datos y se indica al jugador que puede reiniciar la partida porque no se han guardado los datos Se reinicia el objeto partida y sus datos asociados
Operaciones afectadas	--

TABLA 23 MÉTODO CALCULAR Y GUARDAR PUNTUACION

calcularyGuardarPuntuacion(). Servicio de objeto GestorPuntuacion en interfaz con Controlador	
Parámetros de entrada	• ID Partida
Parámetros de salida	• Confirmación de guardado de cálculo y guardado de puntuación o notificación de error
Consideraciones de implementación	Si no se pueden guardar los datos por algún problema, se sale de la partida sin guardar los datos y se indica al jugador que puede reiniciar la partida porque no se han guardado los datos

	Se reinicia el objeto partida y sus datos asociados
Operaciones afectadas	--

TABLA 24 MÉTODO OBTENER RESPUESTA

obtenerRespuestas(). Servicio de objeto GestorRespuestas en interfaz con GestorPuntuacion	
Parámetros de entrada	• ID Partida
Parámetros de salida	• Lista de respuestas con sus tiempos
Consideraciones de implementación	--
Operaciones afectadas	--

TABLA 25 MÉTODO OBTENER RESPUESTAS

obtenerRespuestas(). Servicio de objeto GestorRespuestas en interfaz con GestorPuntuacion	
Parámetros de entrada	• ID Partida
Parámetros de salida	• Lista de respuestas con sus tiempos
Consideraciones de implementación	--
Operaciones afectadas	--

TABLA 26 MÉTODO INDICACION PARTIDA SESION TERMINADA

indicacionPartidaSesionTerminada(). Servicio de objeto SesionBroker en interfaz con Controlador

Parámetros de entrada	• ID Partida • Sesión asociada
Parámetros de salida	• Confirmación de cambio de guardado de registro de partida
Consideraciones de implementación	--
Operaciones afectadas	--

3.3.2.3 DESPLIEGUE DE LA APLICACIÓN

El diseño hecho muestra la estructura y comportamiento del sistema pero para finalizar de ver el sistema en su entorno se tiene que tener en cuenta el contexto donde se va a desplegar la aplicación. Esto quiere decir que los requerimientos no funcionales definidos en la fase de comunicación tienen cierto efecto en la implementación, por lo tanto hay que realizar un diagrama de despliegue para definir las interfaces de comunicación y protocolos para aplicaciones distribuidas que es una parte relevante del sistema.

Los requerimientos no funcionales que tienen influencia en el despliegue son los siguientes:

- 2-rnf-SO: La aplicación es para el sistema operativo Android. Esto implica que la aplicación es sólo para dispositivos móviles
- 7-rnf: La gestión del contenido y los usuarios se realiza desde la institución que tiene derechos de distribución sobre la aplicación. Esto implica que la institución tiene un operador de la aplicación que gestiona el acceso y el contenido de la extensión de ILLLab.
- 5-rnf: La aplicación debe ser compatible con el estándar de intercambio de datos con fines educativos Common Cartridge. Esto implica la definición de un componente que tenga una interfaz de comunicación que sea compatible con este estándar.

Diagrama de despliegue

El despliegue de la aplicación ILLLab incluye dos nodos principales que son el servidor desde donde se accede al contenido y a los datos para que la aplicación funcione y el dispositivo móvil que incluye la interfaz gráfica. Luego, dentro del nodo del servidor se encuentran los componentes de base de datos

(MySQL database), herramienta de gestión de bases de datos (MySQLWorkbench, gestor de contenido (Exe Content Editor) y el sistema de ficheros donde se suban los ficheros del contenido generado por el gestor de contenido.

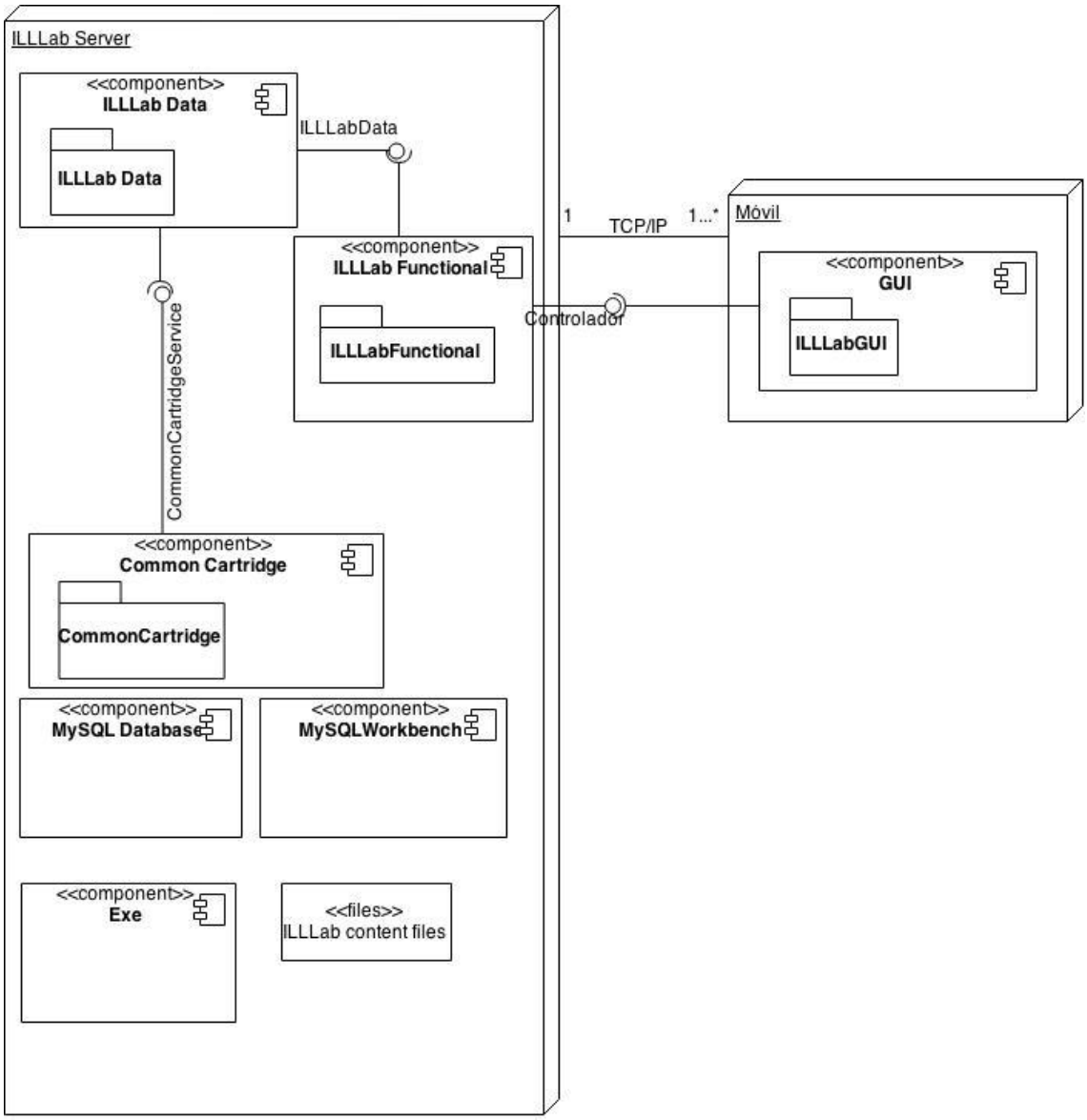


ILUSTRACIÓN 29 DIAGRAMA DE DESPLIEGUE

La base de datos junto con la herramienta de gestión pueden estar distribuidos por la red pero en este caso están en el mismo PC por lo tanto sólo se comunican por un puerto TCP. Luego el gestor de contenidos puede estar alojado en cualquier ordenador sólo que el contenido se debe descargar en el mismo ordenador donde esté el servidor Tomcat. En el caso de que no se despliegue en el mismo servidor se tendrá que agregar un cliente FTP con conexión segura al código en el componente CommonCartridge para acceder a los ficheros.

Las siguientes interfaces surgen de los componentes a desplegar:

Interfaz ILLLabData

- nuevaPartida()
- obtenerPregunta()
- obtenerRestpuesta1()
- ..
- obtenerRespuestaN()
- obtenerDatosPartida()
- obtenerReferenciaContenido()
- obtenerListaJugadores()
- comprobarJugador()
- iniciarSesion()
- obtenerSesiones()
- SignalConfirmacionSesion()
- inicioPartidaSesion()
- indicacionPartidaSesionTerminada()
- ~~eliminaSesion()~~
- obtenerListaJugadores()

Interfaz Controlador

- listaJugadores()
- invitarJugador()
- Salir()
- Login()
- IndicacionConfirmacionSesion()
- obtenerDatosPartida()
- obtenerRespuestaSeleccionada()
- obtenerPuntuacion()
- ~~borrarSesion()~~
- ActualizarSesiones()

Interfaz CommonCartridgeService

Ésta interfaz se encuentra implementada en el objeto CommonCartridgeService que es una abstracción del estándar para el acceso de datos como si fueran objetos. La especificación no se encuentra en los

diagramas de secuencia debido a que el acceso de datos es inmediato y el modelado de los objetos se encuentra en Java como Strings o Lists.

- getPregunta()
- getRespuestas()
- getRespuestaCorrecta()
- obtenerDatosPartida()
- getNumeroPreguntas()

3.4 DESARROLLO DE LA VISTA

3.4.1 PROYECTO BASE

Este proyecto parte desde la base del proyecto ILLLab, entonces se ha importado el código del proyecto base y luego se ha integrado el código del diseño en el mismo. Para importar el proyecto se debe descargar el proyecto "ILLLabExtension" en el disco duro del PC desde el se esté trabajando y utilizar la herramienta de importación de projectos Java de Eclipse. Así comienza la integración y continua con la inclusión del resto del código en paquetes complementarios al proyecto y en proyectos Java que soportan la comunicación para sistemas distribuidos.

A continuación se presenta una descripción del proceso de programación e integración del trabajo realizado.

Importar el proyecto

Para importar el proyecto se selecciona "Import" del menú deplegable de "File"

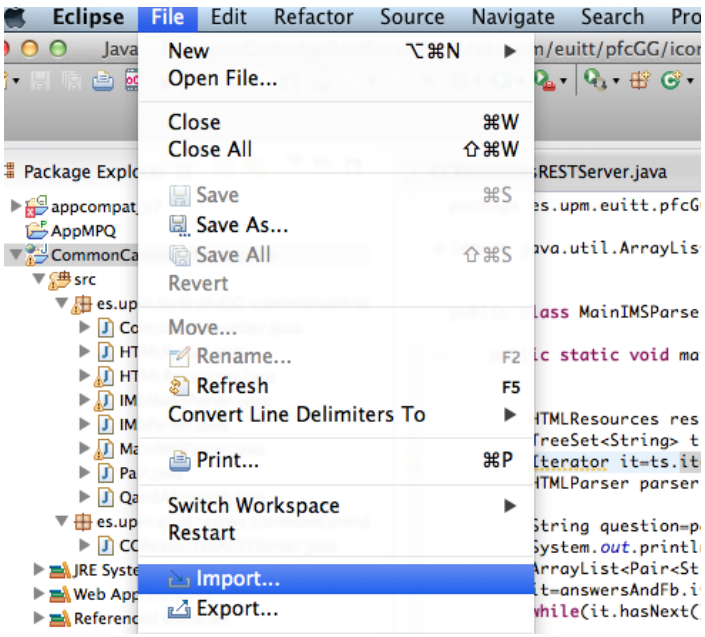


ILUSTRACIÓN 30 IMPORTAR PROYECTO EN ECLIPSE

Luego se procede a importar el proyecto como un proyecto Java

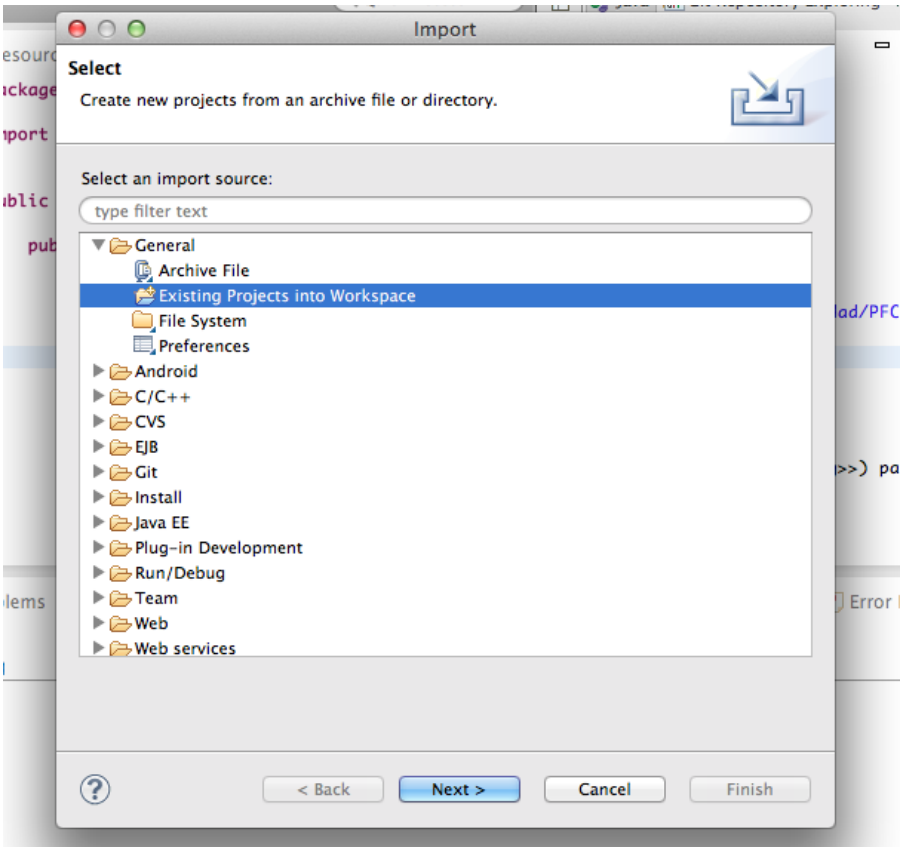


ILUSTRACIÓN 31 IMPORTAR PROYECTO JAVA EN ECLIPSE

Después de seleccionar la opción de "Existing projects into workspace" se abre una ventana para buscar la carpeta donde está el proyecto. Se selecciona la carpeta principal y se ejecuta la importación automáticamente.

Crear clases desde TopCoder UML Tool

Para importar las clases del diseño, es necesario exportar el código desde la herramienta TopCoder UML Tool. Simplemente se abre un proyecto sobre el que se esté trabajando yendo a "File" y luego seleccionando "Open" en el menú desplegable. Se selecciona el proyecto a continuación se selecciona el diagrama de clases que se haya generado. Una vez sobre el diagrama se va al menú y se hace clic sobre la opción "Generate" para luego seleccionar "Generate Code" del menú desplegable (es la única opción disponible).

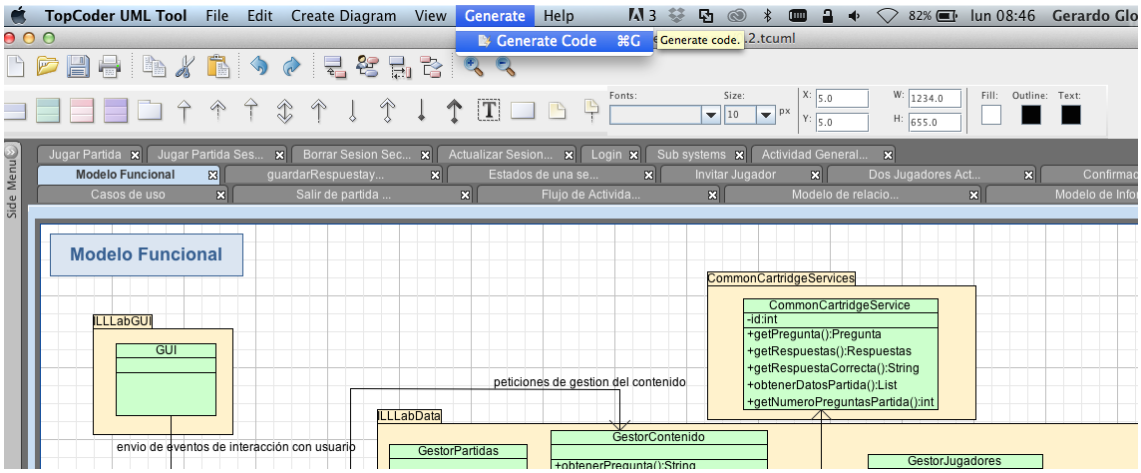


ILUSTRACIÓN 32 GENERACION DE CODIGO EN TOPCODER

Después de haber seleccionado la ventana para generar código a partir del diseño, se abre una ventana para seleccionar la carpeta destino y el lenguaje de programación (Java o C). En este caso se selecciona Java.

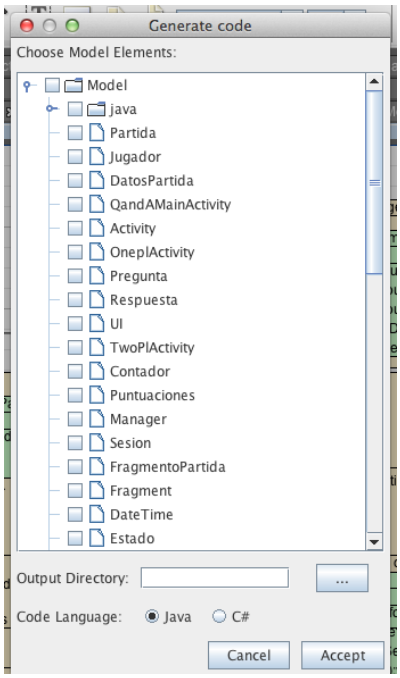


ILUSTRACIÓN 33 SELECCION DE OBJETOS A EXPORTAR

Integrar clases del modelo en el proyecto actual

Después de haber generado las clases y haber importado el proyecto base, fue necesario integrar ambas partes. Para ello se generó una serie de paquetes en el proyecto que integran todas las clases funcionales:

- Es.upm.euitt.ILLLabextensionif
- Es.upm.euitt.ILLLabextensionif.data
- Es.upm.euitt.ILLLabextensionif.functional

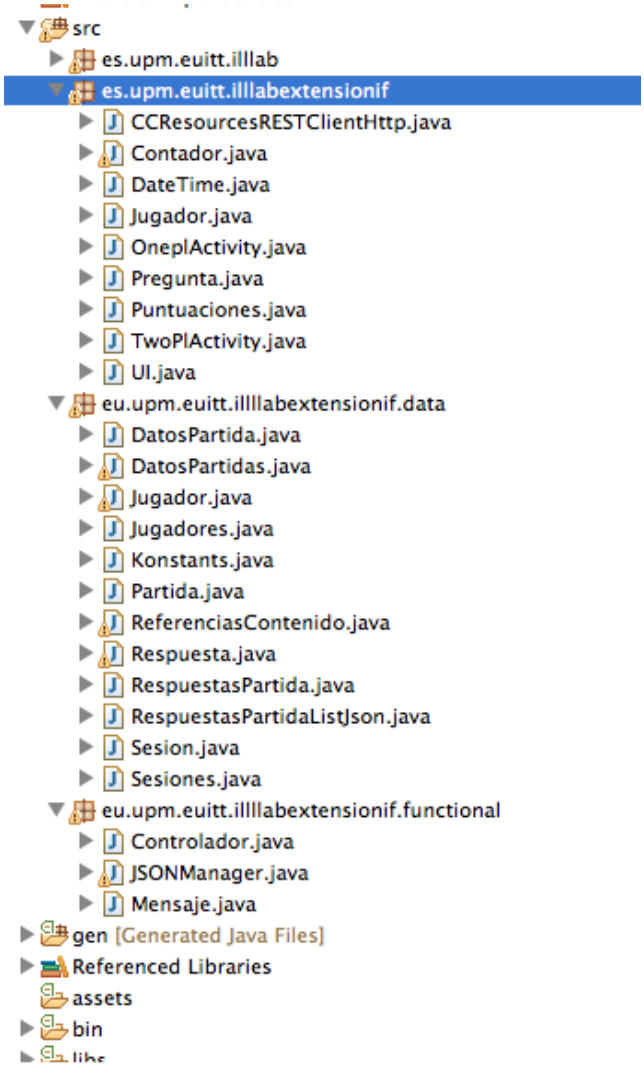


ILUSTRACIÓN 34 PAQUETES DEL PROYECTO ILLLAB

Clases del paquete funcional y de modelo de datos

Dentro del proyecto base, inicialmente sólo se encontraba la aplicación con sus actividades Android correspondientes y todo el contenido para los ejercicios (texto, imágenes, audio, etc). Esto es debido a que la aplicación no estaba planteada como un sistema distribuido [20]. Por el contrario, la extensión de ILLLab permite acceder a todo el contenido a través de Internet. Es por eso que dentro del proyecto se incluyen las clases que permiten gestionar los tipos de datos y las conexiones con el servidor externo que alberga el contenido. El concepto de servidor y su implementación se explican en la sección 3.5.

La clase Controlador es la interfaz principal de conexión al servidor externo. Esta clase actúa como un proxy de la clase Controlador que se encuentra en el servidor. Esto quiere decir que la clase Controlador que se encuentra dentro de la aplicación del dispositivo móvil implementa la interfaz Controlador pero lo único que realiza es la conexión con el servidor y el paso de parámetros a través de Internet al servidor. De esta forma se libera el dispositivo móvil de tener que alojar el contenido a cambio de realizar intercambios de información a través de Internet.

El resto de las clases que se integran en el proyecto base son el soporte para la interpretación y gestión de la información. Éstas clases se encuentran principalmente en el paquete es.upm.euitt.ILLLabextensionif.data y son instancias de objetos del modelo de datos y de los mensajes que se intercambian entre la aplicación y el servidor.

Clases del paquete de la interfaz gráfica

Tal como se indica en [21], las actividades Android constan de la parte lógica y la parte gráfica. La parte lógica está formada por las clases que capturan eventos y la gráfica se compone de ficheros XML asociados a la actividad. Aquí se exponen las actividades desarrolladas para este proyecto. Ambos ficheros, las clases y los archivos XML se han generado con el entorno gráfico de Eclipse que ofrece una serie de herramientas para crear actividades (clases Java y gráficas). A continuación se enseña cómo crear paso a paso una actividad con las herramientas de Eclipse.

Después de haber abierto el projecto ILLLab e integrado las clases del modelo se crean las actividades de la interfaz gráfica. Para crear una actividad con Eclipse simplemente hay que hacer clic derecho en el proyecto Android en el que se está trabajando y hacer clic en “New”, “Other”. Seguido de esta acción se abre un diálogo de opciones donde se debe ir a la carpeta de Android y del menú desplegable se debe elegir Android Activity. En las siguientes figuras ilustra el procedimiento descrito.

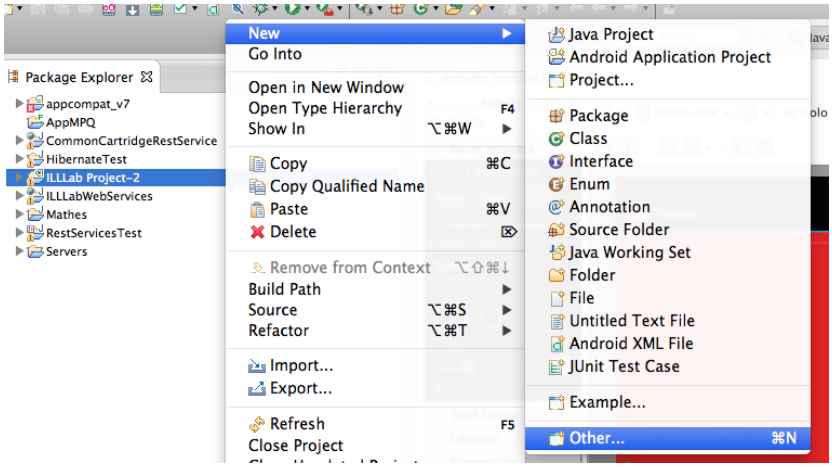


ILUSTRACIÓN 35 CLIC EN NEW-OTHER

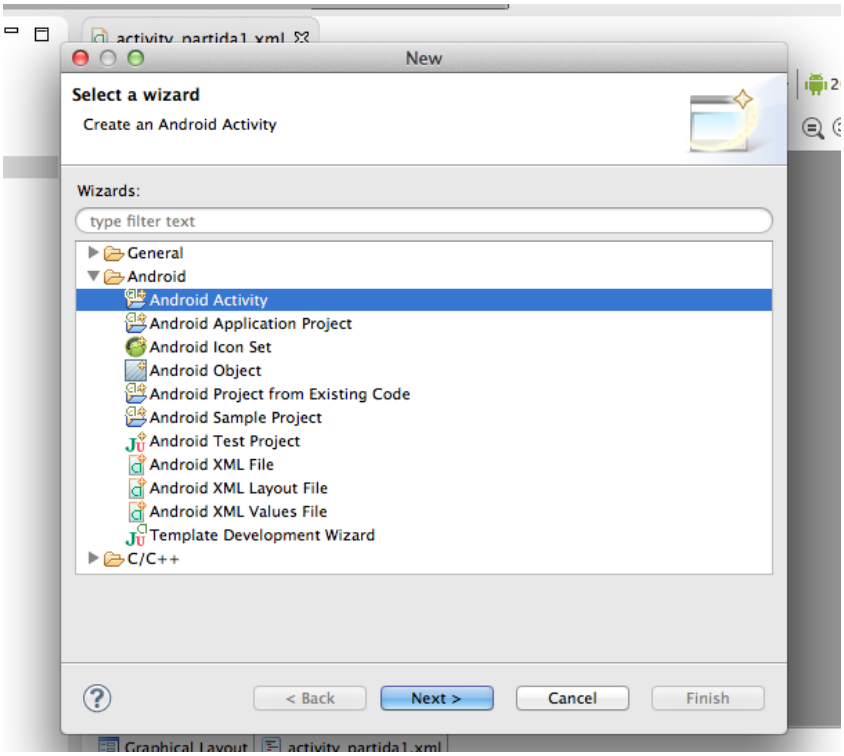


ILUSTRACIÓN 36 ELEGIR ANDROID ACTIVITY DEL MENU DESPLEGABLE

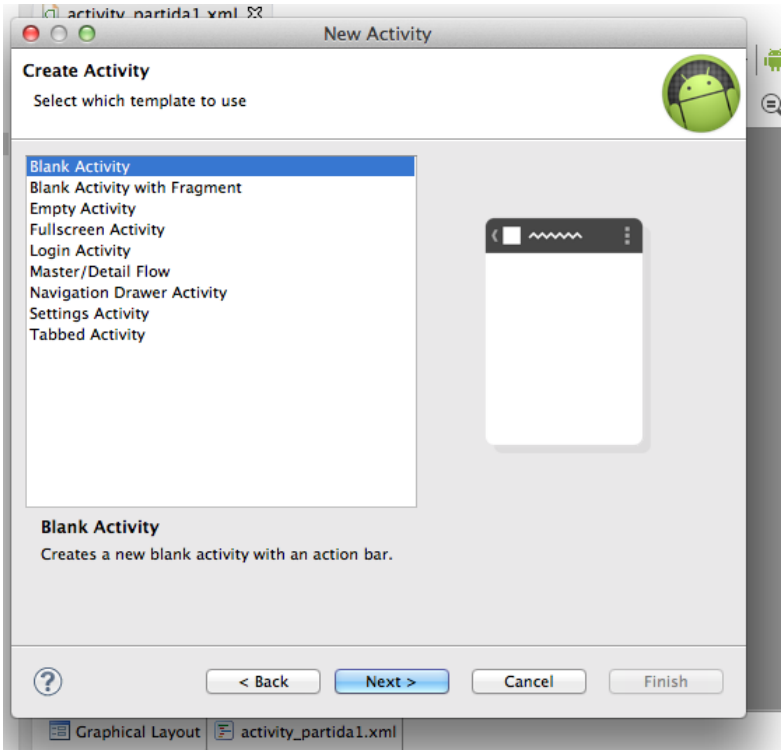


ILUSTRACIÓN 37 ELEGIR BLANK ACTIVITY

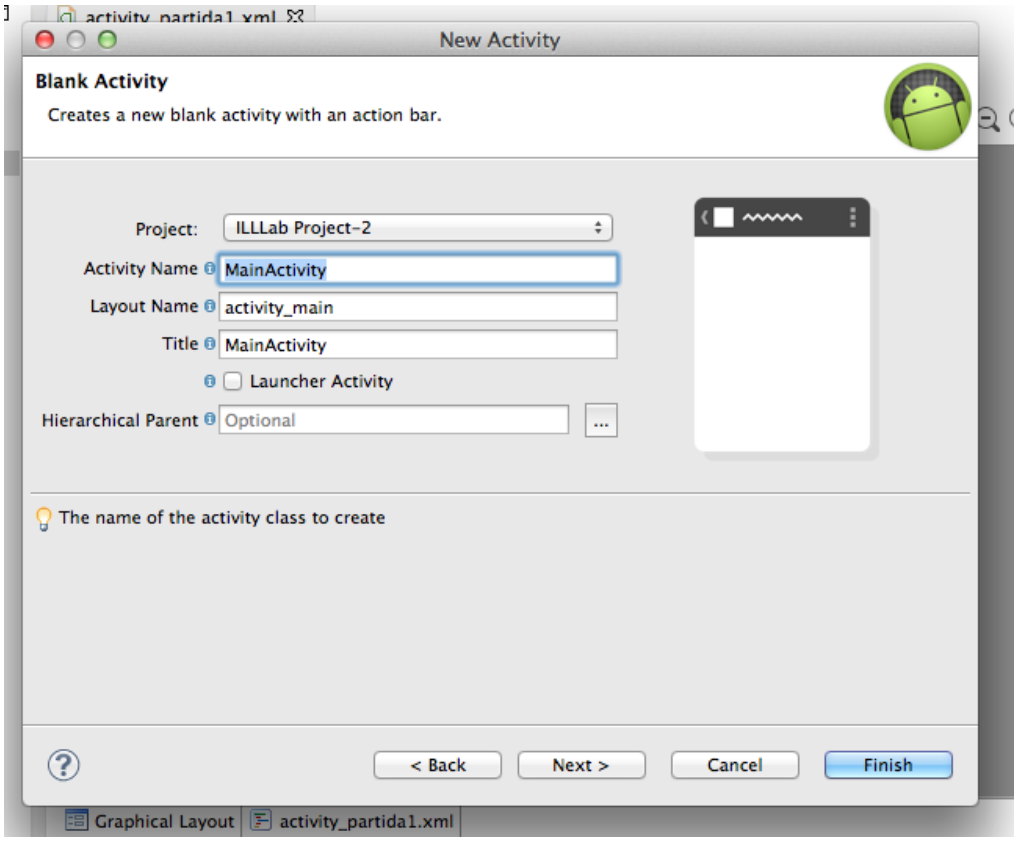


ILUSTRACIÓN 38 ELEGIR UN NOMBRE DE LA ACTIVIDAD

Después de haber hecho clic en "Finish" se crean dos ficheros en el proyecto:

- XXActivity.java: Es la clase Java (XX corresponde al nombre que se le haya atribuido) que captura los eventos, que son producto de la interacción con el usuario, y maneja las instancias de los elementos gráficos desde el punto de vista de la configuración de la vista y posicionamiento en la pantalla. Esta clase se ubica dentro del proyecto java en la carpeta "src"
- Activity_XX.xml: Es el fichero XML que contiene la configuración estática de los elementos de la interfaz gráfica (Layout, Botones, Imágenes, etc). Este fichero se ubica en la carpeta "res->layout".

Para abrir la herramienta de diseño gráfico simplemente es necesario hacer doble clic sobre el fichero .xml que se quiera editar. Para editar la pantalla sólo hace falta seleccionar componentes y arrastrarlos físicamente hacia la pantalla que se presenta como el diseño final de la aplicación que se intenta construir.

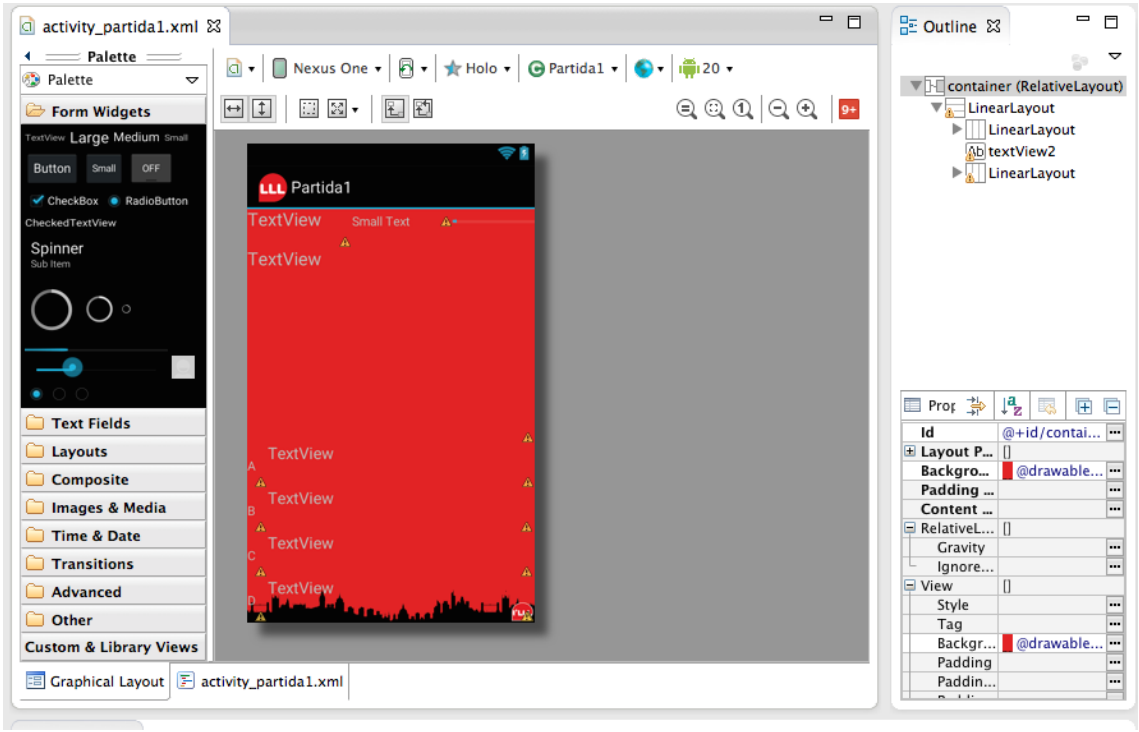


ILUSTRACIÓN 39 HERRAMIENTA DE DISEÑO GRÁFICO DE ECLIPSE

Las clases Java que componen la GUI (Graphical User Interface) están en el paquete es.upm.euitt.ILLLab y se pueden ver integradas en la siguiente imagen.

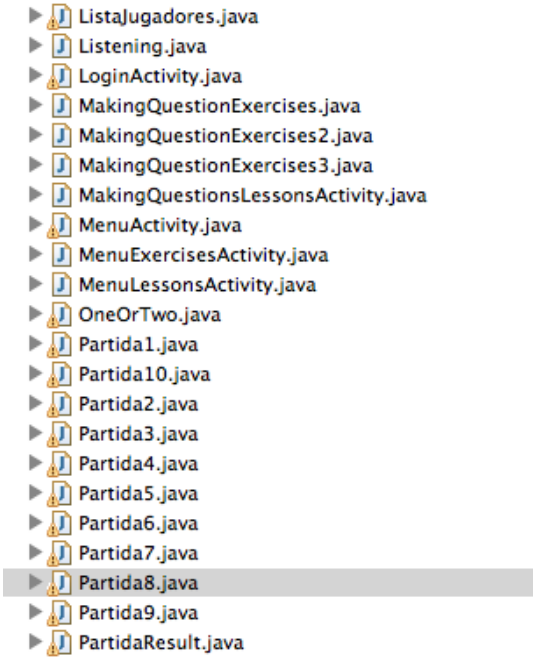


ILUSTRACIÓN 40 CLASES DEL COMPONENTE VISTA (GUI)

Las clases que se generaron son: Partida 1 a 10, PartidaResult, OneOrTwo, LoginActivity, ListaJugadores, SesionesMain, SesionesActivas, SesionActiva, SesionesTerminadas, SesionTerminada y QandAStart. Estas clases forman parte de la parte que captura los eventos generados al interactuar el usuario con el móvil para realizar algún tipo de acción.

A continuación se describen las pantallas que se han diseñado junto con las acciones que puede realizar un usuario de la aplicación.

Pantalla principal

ILUSTRACIÓN 41 CAPTURA DE PANTALLA DE ACTIVIDAD PRINCIPAL EN EMULADOR

Login

ILUSTRACIÓN 42 PANTALLA DE LOGIN EN EMULADOR

Menú Principal

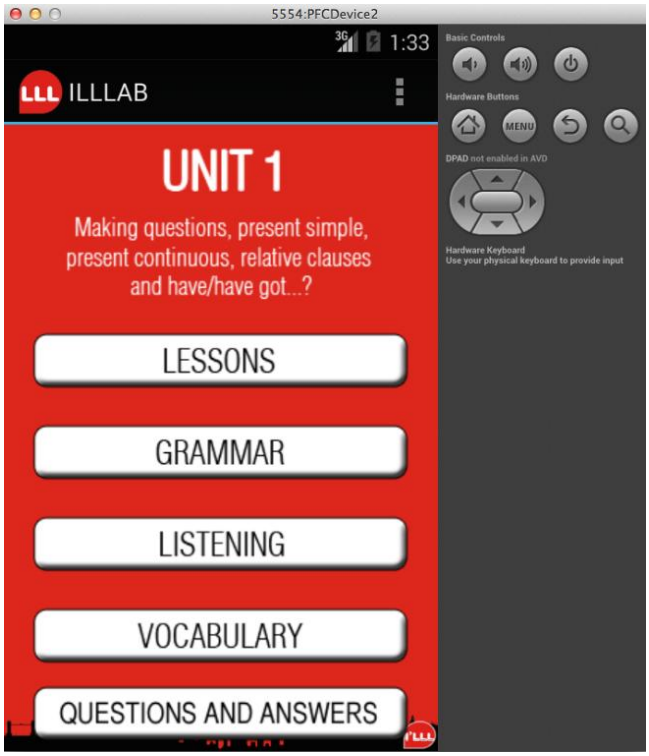


ILUSTRACIÓN 43 PANTALLA DE MENU PRINCIPAL EN EMULADOR

Sección de preguntas y respuestas

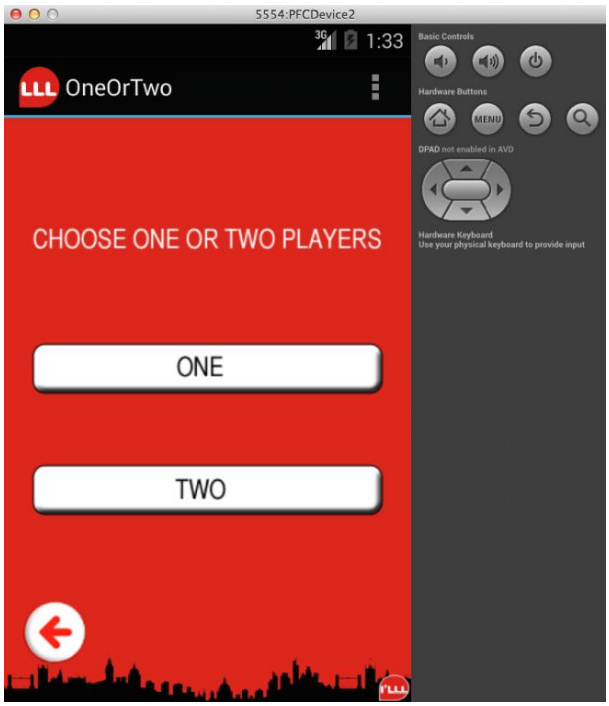


ILUSTRACIÓN 44 PANTALLA DE OPCIONES DE JUEGO EN EMULADOR

Sección de Sesiones de dos jugadores

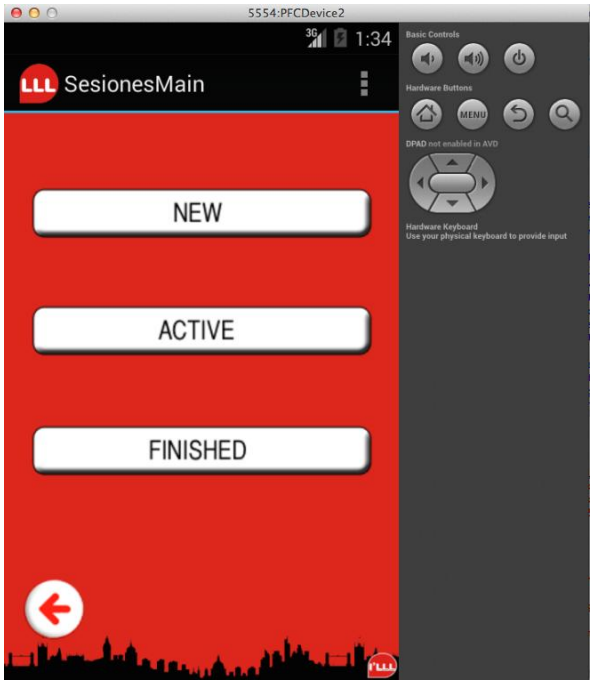


ILUSTRACIÓN 45 PANTALLA DE OPCIONES DE SESION DE DOS JUGADORES EN EMULADOR

Creando nueva sesión con lista de jugadores

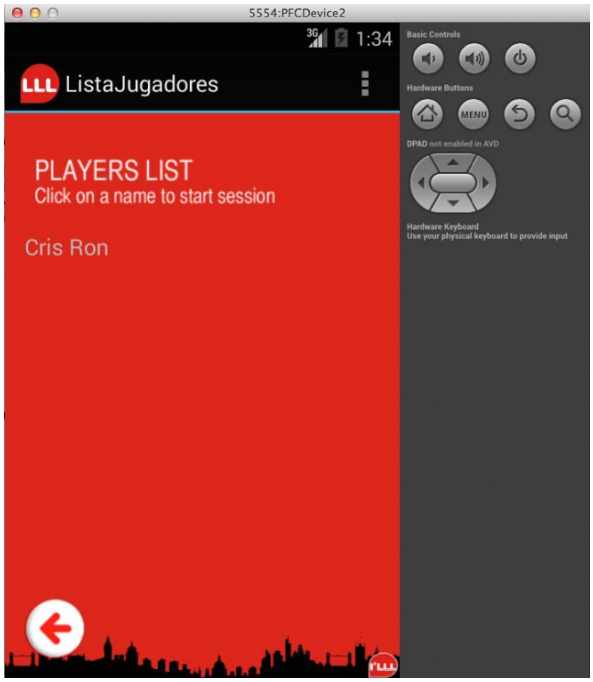


ILUSTRACIÓN 46 PANTALLA DE LISTA DE JUGADORES EN EMULADOR

Sesiones activas

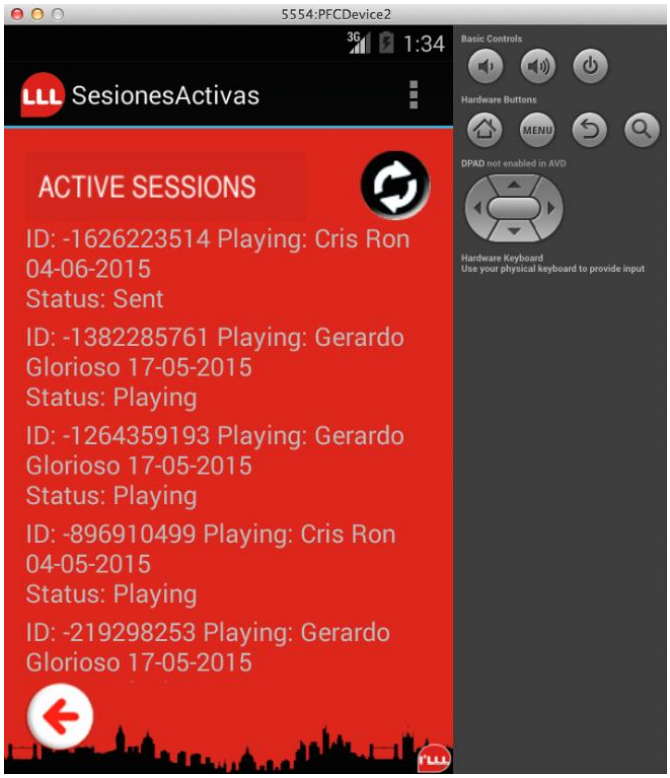


ILUSTRACIÓN 47 PANTALLA DE SESIONES ACTIVAS EN EMULADOR

Sesión recibida

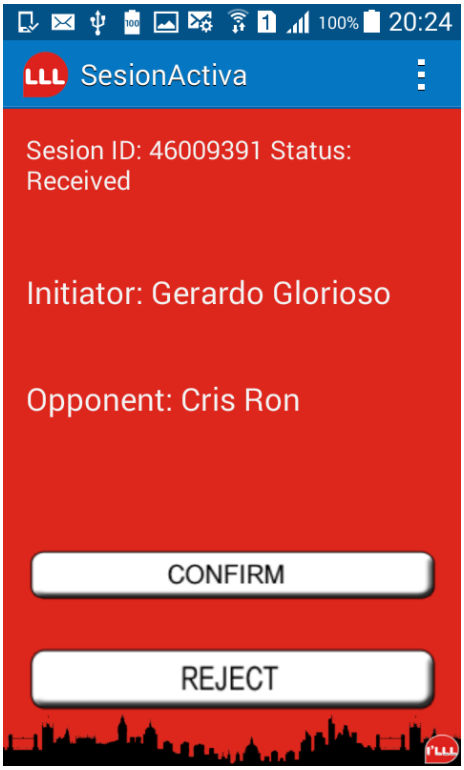


ILUSTRACIÓN 48 PANTALLA DE SESION RECIBIDA EN SAMSUNG GRAND NEO

Sesion Jugada a espera del otro jugador

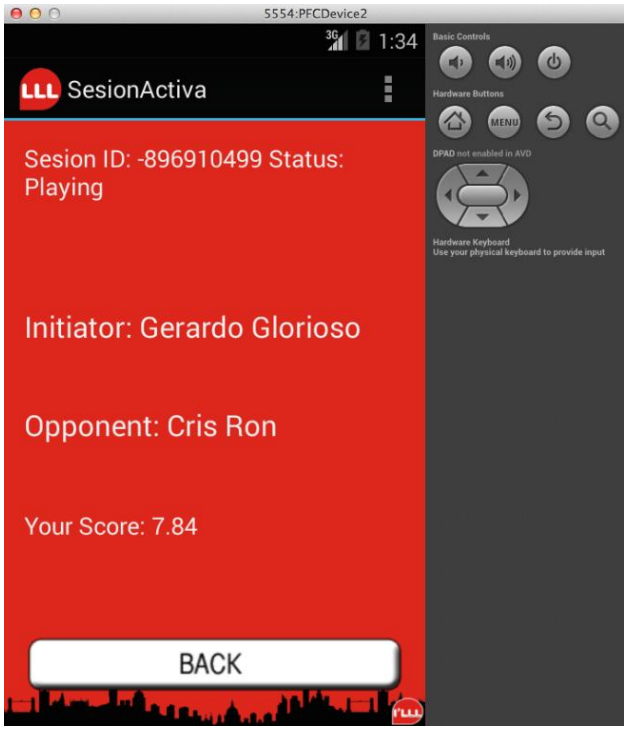


ILUSTRACIÓN 49 PANTALLA DE SESION JUGADA EN EMULADOR

Sesión sin jugar

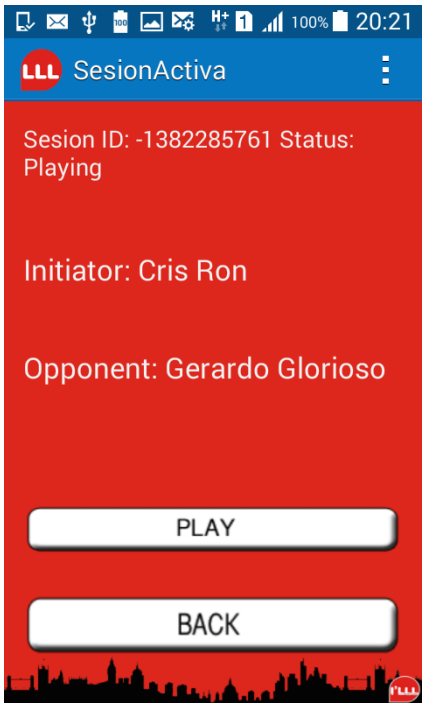


ILUSTRACIÓN 50 PANTALLA DE SESION SIN JUGAR EN SAMSUNG GRAND NEO

Sesiones terminadas

ILUSTRACIÓN 51 PANTALLA DE LISTA DE SESIONES TERMINADAS EN SAMSUNG GRAND NEO

Sesión terminada

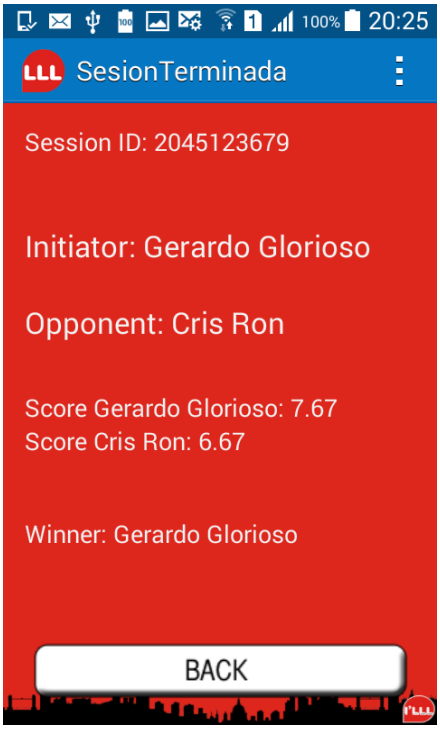


ILUSTRACIÓN 52 PANTALLA DE SESION TERMINADA EN SAMSUNG GRAND NEO

Jugando Partida

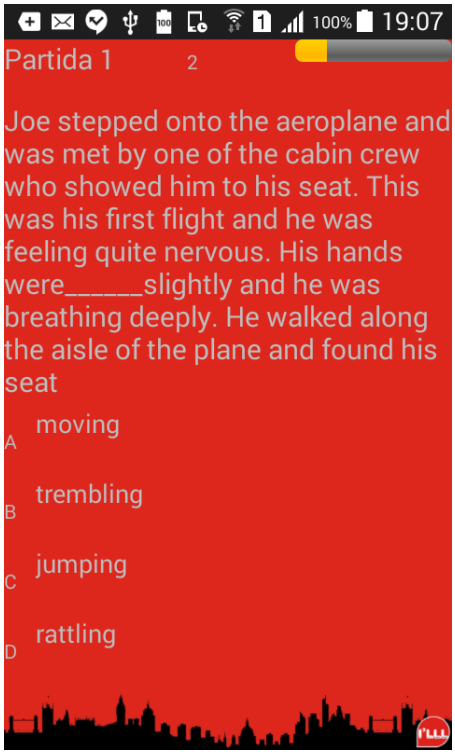


ILUSTRACIÓN 53 PANTALLA DE PARTIDA EN SAMSUNG GRAND NEO

Partida terminada

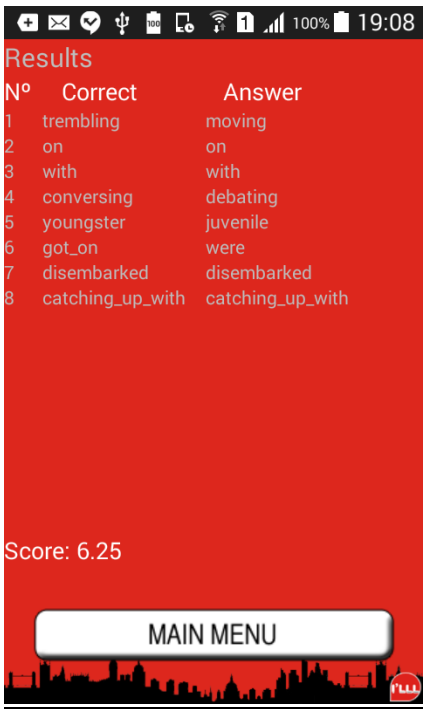


ILUSTRACIÓN 54 PANTALLA DE PARTIDA TERMINADA

3.5 DESARROLLO DEL CONTROLADOR

3.5.1 WEB SERVICE

El proyecto de extensión de ILLLab se realiza en forma de sistema distribuido. Por lo tanto, gran parte de la ejecución de procesos y acceso de información se realiza en un servidor con un Web Service[22] dedicado para realizar las funciones de realización de la lógica de negocio[23] y el acceso al contenido.

En este caso, la clase que da la interfaz a las funcionalidades del sistema es la clase "Controlador". Por lo tanto, en la parte del móvil esta clase es sólo un proxy del objeto que tiene contacto con el contenido y las clases que realizan la lógica de negocio. Esto implica que en la clase controlador se encuentra en dos instancias: (1) en la parte móvil con un cliente de Web Service y en (2) en la parte del servidor como servidor de Web Service. En las siguientes imágenes se observa la diferencia entre las implementaciones del Controlador en la parte móvil y en el servidor.

```
/**
 * @param telefono
 * @return
 */
public List<Jugador> listarJugadores(String telefono) {

    String resp=null;
    resp=cliente.target(Konstants.BASE_URI+Konstants.LISTARJUGADORES_URI+telefono);

    invbuilder= wt.request("application/json");
    response=invbuilder.get();
    resp = response.readEntity(String.class);
    Type t = new TypeToken<Jugadores>() {}.getType();
    Jugadores jugs=(Jugadores) JSONManager.fromJSON(resp,t);
    List<Jugador> list=Arrays.asList(jugs.jugadores);
    return list;
}
```

ILUSTRACIÓN 55 MÉTODO QUE OBTIENE LA LISTA DE JUGADORES DISPONIBLES. PARTE MOVIL

```
/**
 * @param telefono
 * @return
 */
@GET
@Path("/listarJugadores/{telefono}")
@Produces("text/json")
public Response listarJugadores(@PathParam("telefono") String telefono) {


    gjugadores=new GestorJugadores(getSession());


    Jugadores jugadores=gjugadores.obtenerListaJugadores(telefono);
    String cs=JSONManager.toJSON(jugadores);

    return Response.status(200).entity(cs).build();
}
```

ILUSTRACIÓN 56 MÉTODO QUE OBTIENE LA LISTA DE JUGADORES DISPONIBLES. PARTE DEL SERVIDOR

Para montar un Web Service es necesario tener un servidor Web y en este caso se eligió Apache Tomcat [24]. Para desarrollar y desplegar un Web Service REST en Tomcat se deben realizar los siguientes pasos:

1. Crear un proyecto Web en Eclipse (Dynamic Web Project)
2. Desarrollar el proyecto
3. Configurar el fichero web.xml
4. Exportar el proyecto como fichero WAR
5. Desplegar en servidor Tomcat

Para realizar (1) simplemente se debe seguir las instrucciones en [25]. Además se deben recalcar los siguientes detalles:

- A la hora de crear el proyecto se debe indicar en “Context Root”: ILLLabWebServices. Esto indica el nombre con el que se registra el Web Service en el servidor Tomcat y por lo tanto es lo que va en la URL que llama al servicio

- La configuración de web.xml es la siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID"
version="2.5">
  <display-name>ILLLabWebServices</display-name>
      <servlet>
        <servlet-name>JAX-RS Servlet</servlet-name>
        <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
        <init-param>
            <param-name>jersey.config.server.provider.packages</param-name>
            <param-value>eu.upm.euitt.ILLLabif.functional</param-value>
      </init-param>
      <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>JAX-RS Servlet</servlet-name>
        <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>
</web-app>
```

ILUSTRACIÓN 57 CONFIGURACION TOMCAT

La parte del desarrollo se lleva a cabo introduciendo código en las clases que se han generado en la fase de diseño. En este proyecto específicamente se han creado tres paquetes:

- eu.upm.euitt.ILLLabif.commonCartridge
- eu.upm.euitt.ILLLabif.data
- eu.upm.euitt.ILLLabif.data.errors
- eu.upm.euitt.ILLLabif.functional

El paquete eu.upm.euitt.ILLLabif.commonCartridge contiene métodos para acceder al contenido para la aplicación. La clase CCServices permite acceder directamente a las preguntas y respuestas de cada partida. Esta clase es la que se expone a la aplicación ILLLab del servidor, el resto realizan las funciones para explorar el fichero IMS Manifest [referencia a párrafo] (IMSManifestParser e IMSParser), parsear el contenido que se encuentra en HTML (HTMLResources y HTMLParser) y proveer funciones auxiliares para el manejo del contenido (Parser, QandAContent y QuestionsAndAnswersPartida).

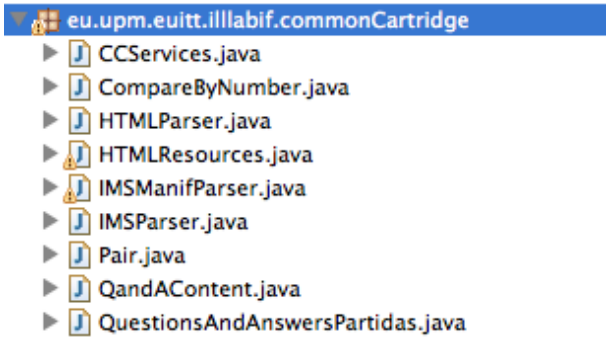


ILUSTRACIÓN 58 PAQUETE COMMONCARTRIDGE

El paquete eu.upm.euitt.ILLLabif.data contiene las clases que representan el modelo de datos (Jugador, Partida, ReferenciasContenido, RespuestasPartida, Sesion) y las listas de éstos objetos que manejan los gestores de datos de la aplicación (RespuestasPartidaListJson, Jugadores, Sesiones).

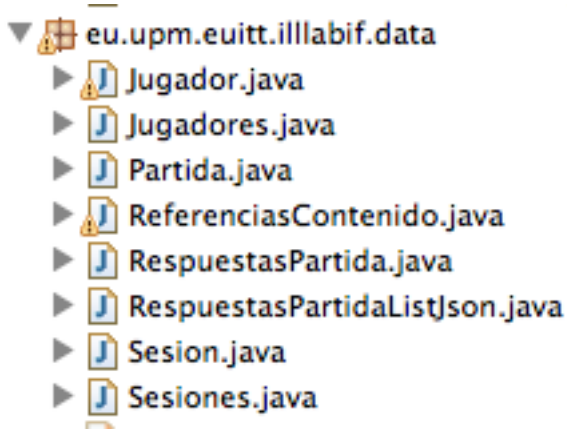


ILUSTRACIÓN 59 PAQUETE EU.UPM.EUITT.ILLLABIF.DATA

El paquete eu.upm.euitt.ILLLabif.data.errors contiene clases que gestionan las excepciones que pueden surgir en respuesta de los errores del sistema y la base de datos. En la siguiente imagen se observan las clases que forman parte del paquete.

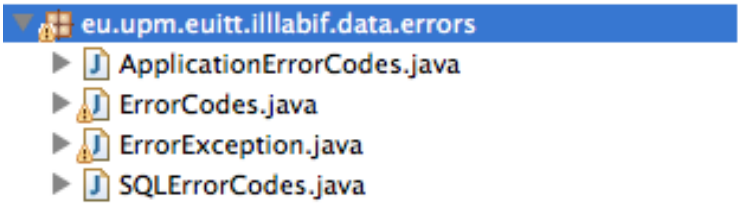


ILUSTRACIÓN 60 PAQUETE EU.UPM.EUITT.ILLLABIF.DATA.ERRORS

El paquete más importante del Web Service es eu.upm.euitt.ILLLabif.functional. Incluye las clases que gestionan la comunicación con el cliente (aplicación en dispositivo móvil), el sistema de ficheros (a través de la interfaz Common Cartridge) y la base de datos. Además, contiene clases que realizan la lógica de negocio y otras que permiten gestionar e interpretar los mensajes que se intercambian en formato Json.

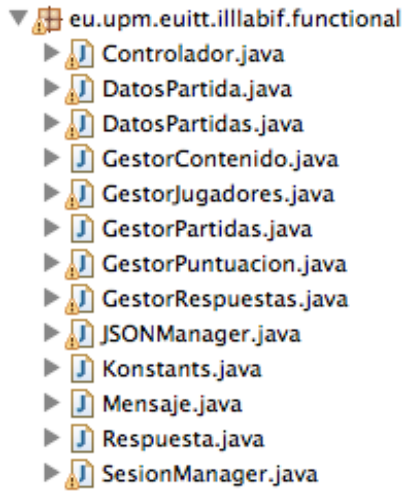


ILUSTRACIÓN 61 EU.UPM.EUITT.ILLLABIF.FUNCTIONAL

Para que funcionen todas estas clases es necesario incluir en el "classpath" todas las librerías de las que dependen las clases[26]. Estas librerías incluyen mayormente el manejo de ficheros HTLM, XML, librerías REST y drivers de bases de datos. Todas las librerías están incluidas en el CD que viene con esta memoria en formato digital.

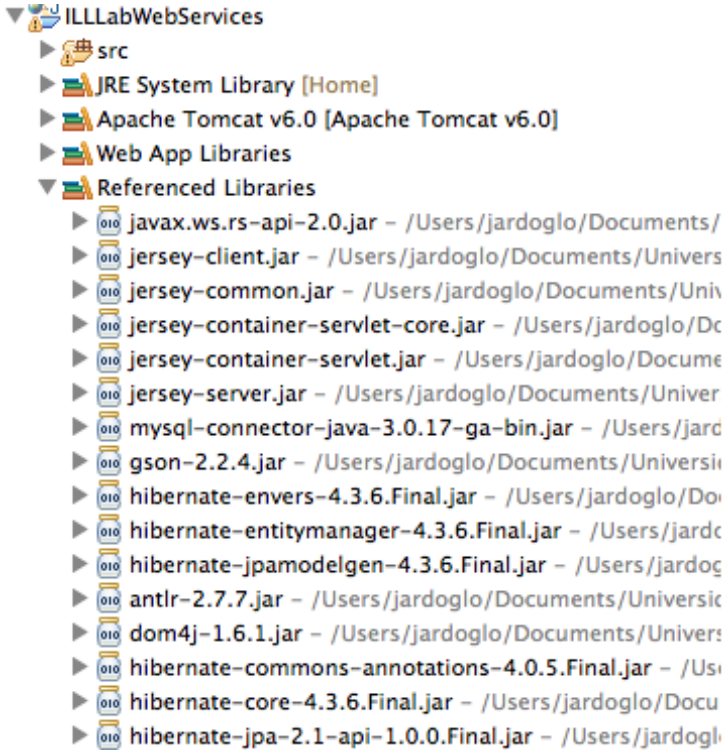


ILUSTRACIÓN 62 RESULTADO DE AGREGAR LIBRERIAS EN EL CLASSPATH DEL PROYECTO

Un aspecto importante a mencionar aquí es cómo se implementa el cálculo de la puntuación final. El juego consta de 10 o menos partidas. Por lo que una respuesta puede valer como máximo 1, en caso de que hayan 10 partidas y $p = 10/n$ si el juego consta de n partidas, donde p es un coeficiente por el cual se

multiplica la calificación final de una respuesta. Se puede decir también que p es un factor de peso de la respuesta.

Por supuesto, diseñar un juego con menos de 6 partidas no tiene sentido, debido a que a medida de que se disminuyen las partidas se debería incrementar la complejidad del texto que en general se traduce en redactar párrafos más extensos para dificultar la lectura. En el caso de una aplicación móvil, el extender el texto no siempre es lo más adecuado debido a la limitación impuesta por las dimensiones de la pantalla.

Cada partida en la secuencia de partidas de un juego consta de una pregunta, cuatro respuestas y una barra de tiempo que cuenta hasta diez segundos. Entonces, el valor de la respuesta esta condicionado por dos factores: (1) el tiempo en que se tarda en contestar (t) y (2) si la respuesta es correcta o no.

Este comportamiento se puede modelar de forma linear o no. En caso de que sea linear, se utiliza la ecuación de una recta para calcular el incremento del tiempo vs. la puntuación. En este caso tenemos la función puntuación de cada partida $c_{(t)}$ y el criterio de evaluación con respecto al tiempo es:

$$c_{(t)}\begin{cases} 1\ ;\ 0 \le t \le a \\ -\dfrac{1}{b-a} \times (t-a) + 1;\ a < t \le b \end{cases} \qquad (1)$$

La variable a corresponde con el tiempo límite antes de que, en caso de que la respuesta sea correcta, el puntaje empiece a disminuir. La variable b es el tiempo límite para responder una pregunta. Después de ese límite la puntuación es cero. La siguiente imagen corresponde a la función $c_{(t)}$ representada en dos funciones $f_{(x)}\ y\ \ g_{(x)}$ cuando $a = 4\ y\ b = 10$ (representadas en el intervalo 0<x<10). En este caso $c_{(t)}$ es $f_{(x)}$ en el intervalo [0,4] y $g_{(x)}$ en el intervalo (4,10].

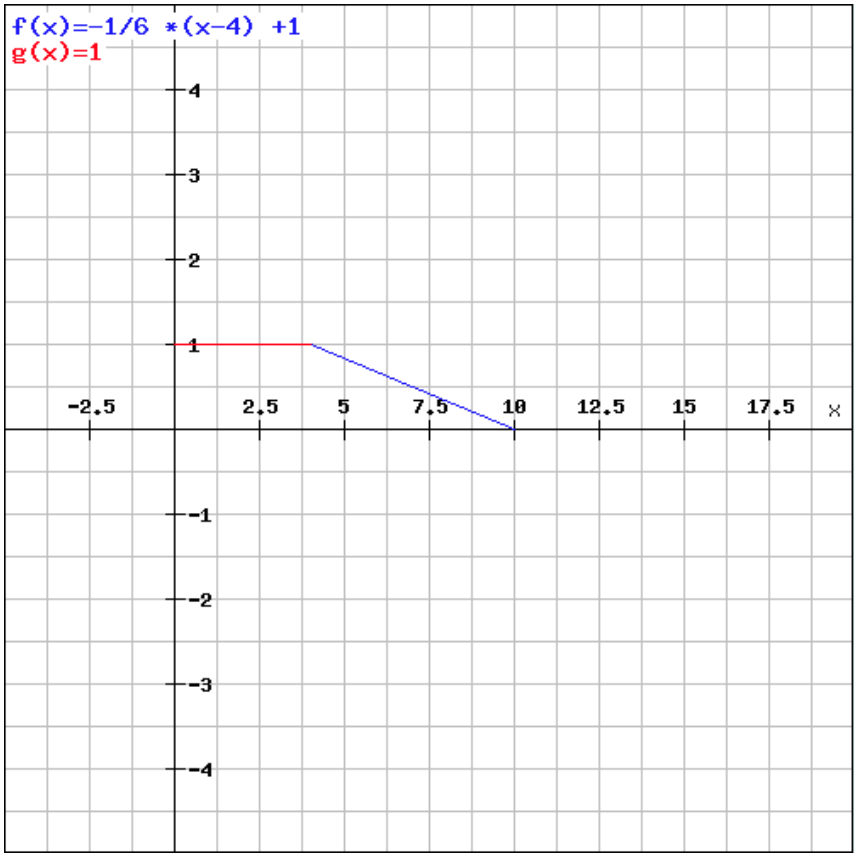


ILUSTRACIÓN 63 FUNCION C(T) EN DOS FUNCIONES: F(X) Y G(X)

La puntuación final $S_{(n,t)}$ entonces es la suma de las puntuaciones de cada partida multiplicada cada una por el factor de peso que depende de la cantidad de partidas que haya en el juego.

$$S_{(n,t)} = \frac{10}{n}\sum_{1}^{i} c_{i(t)} \quad (2)$$

Esto luego se ha codificado en Java dentro de la clase que gestiona las puntuaciones (GestorPuntuacion).

```
private String calcularPuntuacion(List<?> respuestas) {

		Iterator<?> itResp;

		itResp=(Iterator<?>) respuestas.iterator();

		Float puntuacion=(float) 0;

		while(itResp.hasNext()){

			Object a=itResp.next();

			RespuestasPartida re=(RespuestasPartida) a;
			Float parametro=(float) respuestas.size();
			Float tiempo=new Float(re.tiempo);
			Float f=new Float(0);
			Float factor=10/parametro;
			if(re.respuesta.equalsIgnoreCase(re.correcta)){

				if(new Float(tiempo)<=4){
					f=(float)factor;
				}else{
					f=(float) 1-(float)((0.167)*(tiempo-4));
				}
			}
			puntuacion+=(f*factor);
		}
		BigDecimal big=new BigDecimal(puntuacion.toString());
		big=big.setScale(2, RoundingMode.HALF_UP);

		return big.toString();
	}
```

ILUSTRACIÓN 64 CODIGO QUE CALCULA LA PUNTUACION DE CADA PARTIDA

3.6 DESARROLLO DEL MODELO

La base de datos esta gestionada mediante la herramienta MySQLWorkbench y el acceso mediante código con el framework Hibernate. En esta sección se indica cómo se ha configurado la base de datos y el acceso mediante las herramientas de Hibernate para Eclipse.

3.6.1 CONFIGURACIÓN Y DESPLIEGUE DE BASE DE DATOS

Configuración de MySQLWorkbench

Antes de utilizar la herramienta de gestión de la base de datos se debe configurar mysql. Para descargar mysql se debe entrar en http://www.mysql.com/downloads/, ir a la sección de MySQL Community

Edition (GPL), descargar MySQL Community Server e instalar el fichero que corresponda con el instalador del sistema operativo sobre el que se esté trabajando.

Luego de haber instalado mysql se abre una ventana para configurar la contraseña para el usuario "root". Para esto se deben seguir las instrucciones dadas en [27]. Finalmente, mediante el comando *"sudo mysql –-user=root --password"* se accede al servidor como administrador.

Después de haber configurado mysql se accede a la herramienta MySQLWorkbench para crear el modelo de la base de datos y desplegarlo en el servidor.

Creación de base de datos

Para crear el modelo se va al menú de opciones de MySQLWorkbench y se selecciona File->New Model. Una vez que se abre la ventana para gestionar el modelo se hace clic sobre el icono "Add Diagram" para crear un nuevo modelo.

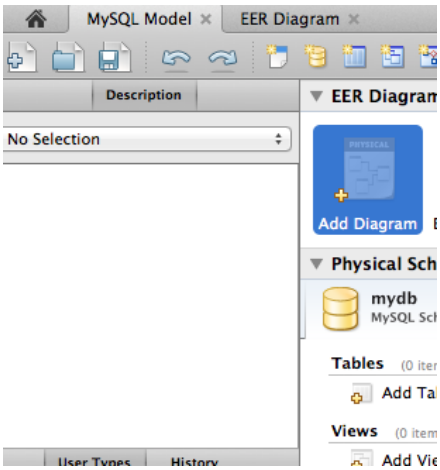


ILUSTRACIÓN 65 MYSQL WORKBENCH. ADD DIAGRAM

Al crear un nuevo diagrama se abre un editor de diagramas EER (Enhanced entity-relationship model) donde se puede empezar a diseñar el modelo de datos. En este caso se ha diseñado un diagrama acorde con el modelo de datos creado en la fase de diseño de la aplicación. El modelo se muestra en la siguiente figura.

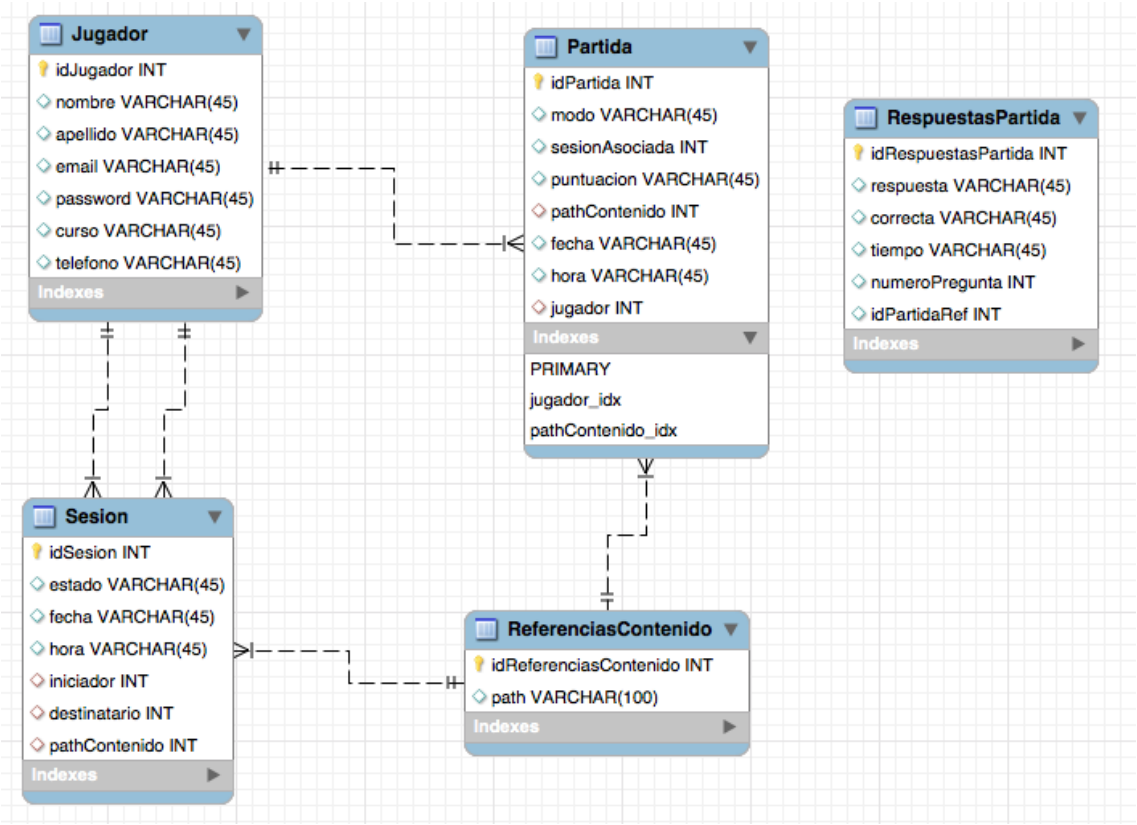


ILUSTRACIÓN 66 MODELO DE DATOS MYSQL

Para desplegar el modelo es necesario ejecutar el script SQL que se genera al diseñar el diagrama EER. Eso se consigue yendo al menú principal, se selecciona Database y del menú desplegable se hace clic en "Forward Engineer…".

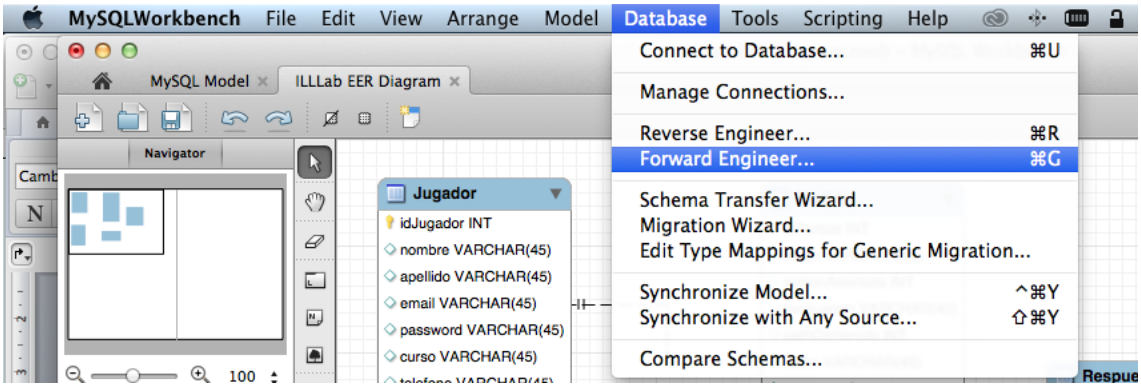


ILUSTRACIÓN 67 MYSQL WORKBENCH. FORWARD ENGINEER

Al seleccionar "Forward Engineer…" se abren un a serie de diálogos para seleccionar la conexión, los comandos SQL para desplegar la base de datos y los elementos que se desean desplegar.

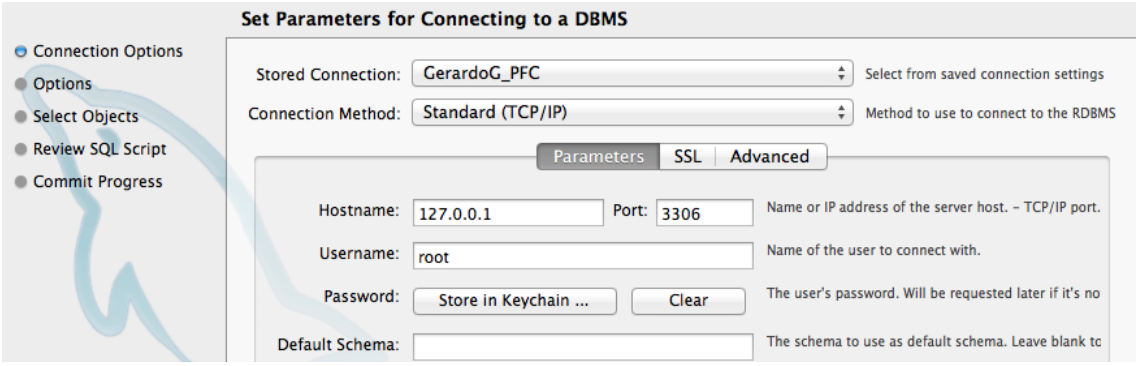


ILUSTRACIÓN 68 CREAR BASE DE DATOS PASO 1

En el dialogo donde pide seleccionar las opciones de despliegue se debe elegir "DROP Objects Before Each CREATE Object" y "Generate DROP SCHEMA" tal como indica en la siguiente imagen. Después pide la contraseña de acceso al servidor MySQL que es la misma que se ha puesto para el usuario root.

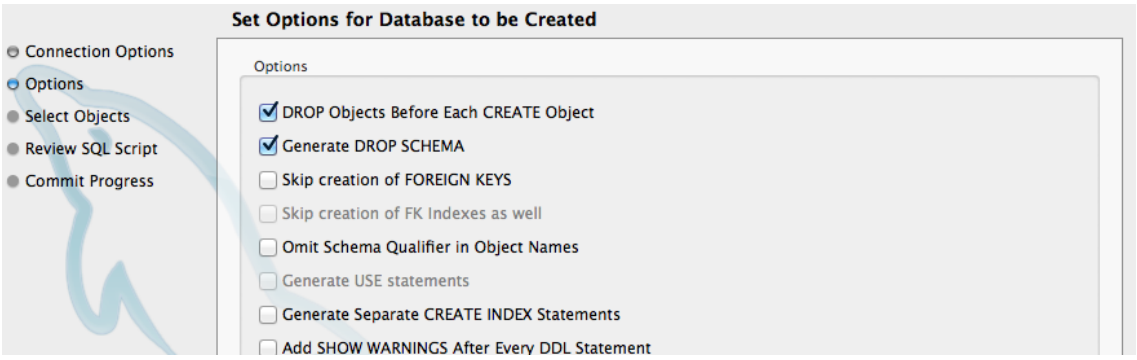


ILUSTRACIÓN 69 CREAR BASE DE DATOS PASO 2

En el diálogo para seleccionar el tipo de objetos a desplegar se elije "Export MySQL Table Objects" y se selecciona "continuar" hasta que se despliegue la base de datos.

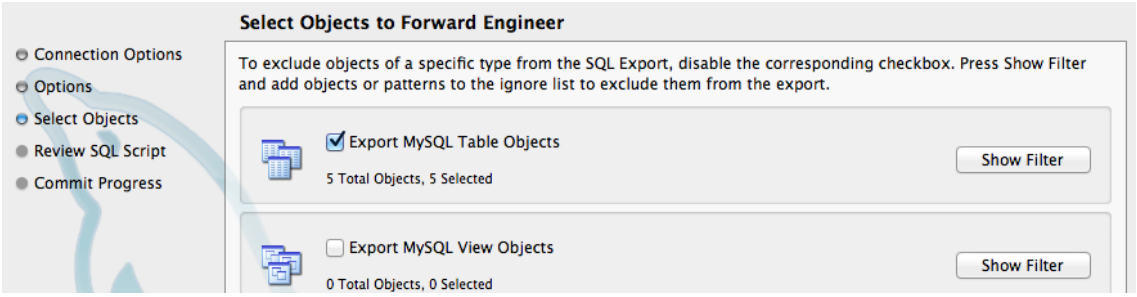


ILUSTRACIÓN 70 CREAR BASE DE DATOS PASO 3

Configuración de Hibernate en Eclipse

Hibernate es un framework que permite manejar los objetos de la base de datos mysql como si fueran objetos Java. Para esto es necesario configurar la conexión de Hibernate con la base de datos y crear ficheros de mapeo de objetos de la base de datos con los objetos Java.

Para realizar la configuración de Hibernate es necesario seguir los pasos que se indican en [28]. En este caso hay que tener en cuenta los siguientes detalles que son distintos de la configuración que se indica en [29]: el conector de la base de datos que se utiliza es el de mysql, los ficheros de configuración y mapeo ya vienen integrados en el proyecto y las librerías que hay que agregar se encuentran entre las librerías que hay que agregar en el proyecto ILLLabWebServices.

Al final el proyecto debe incluir el fichero de configuración bajo la carpeta "src" del proyecto ILLLabWebServices y debe incluir los parámetros que se muestran en la siguiente figura.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
            "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
            "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.password">root</property>
        <property
name="hibernate.connection.url">jdbc:mysql://127.0.0.1:3306/mydbPfc</property>
        <property name="hibernate.connection.username">root</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

        <mapping resource="Jugador.hbm.xml"/>
        <mapping resource="Partida.hbm.xml"/>
        <mapping resource="ReferenciasContenido.hbm.xml"/>
        <mapping resource="RespuestasPartida.hbm.xml"/>
        <mapping resource="Sesion.hbm.xml"/>


        <mapping class="eu.upm.euitt.ILLLabif.data.Jugador"/>
        <mapping class="eu.upm.euitt.ILLLabif.data.Partida"/>
        <mapping class="eu.upm.euitt.ILLLabif.data.ReferenciasContenido"/>
        <mapping class="eu.upm.euitt.ILLLabif.data.RespuestasPartida"/>
        <mapping class="eu.upm.euitt.ILLLabif.data.Sesion"/>

    </session-factory>
</hibernate-configuration>
```

ILUSTRACIÓN 71 CONFIGURACION FICHERO CFG HIBERNATE

Las clases Java que tienen ficheros de mapeo asociados son las que representan el modelo de datos desplegado en la base de datos: Jugador, Sesión, Partida, ReferenciaContenido y RespuestasPartida. Los ficheros de mapeo deben incluir las correspondencias entre los campos de las clases Java y objetos de la base de datos y el tipo de datos. En la siguiente imagen se encuentra un ejemplo de cómo se escribe un

fichero de mapeo. En este caso es el fichero que corresponde al mapeo de la clase Jugador y el objeto Jugador de la base de datos.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated 13-feb-2015 0:02:06 by Hibernate Tools 3.4.0.CR1 -->
<hibernate-mapping>
    <class name="eu.upm.euitt.ILLLabif.data.Jugador" table="JUGADOR">
        <id name="idJugador" type="int" access="field">
            <column name="IDJUGADOR" />
            <generator class="assigned" />
        </id>
        <property name="nombre" type="java.lang.String" access="field">
            <column name="NOMBRE" />
        </property>
        <property name="apellido" type="java.lang.String" access="field">
            <column name="APELLIDO" />
        </property>
        <property name="email" type="java.lang.String" access="field">
            <column name="EMAIL" />
        </property>
        <property name="password" type="java.lang.String" access="field">
            <column name="PASSWORD" />
        </property>
        <property name="curso" type="java.lang.String" access="field">
            <column name="CURSO" />
        </property>
        <property name="telefono" type="java.lang.String" access="field">
            <column name="TELEFONO" />
        </property>
    </class>
</hibernate-mapping>
```

ILUSTRACIÓN 72 CONFIGURACION MAPEO HIBERNATE

Después de configurar Hibernate, es posible manejar objetos de la base de datos como si fueran objetos Java. Hibernate da la posibilidad de actualizar objetos y de realizar consultas similares a las de SQL. En las siguientes imágenes se pueden ver dos ejemplos de manejo de objetos con Hibernate.

```
public String obtenerReferenciaContenido() {

       Query query = session.createQuery("select path from ReferenciasContenido
order by rand() lim 1");

       List<?> list = query.list();
       String pathContenido=(String)list.get(0);

         return pathContenido;
    }
```

ILUSTRACIÓN 73 EJEMPLO DE CONSULTA HIBERNATE

```
	public Mensaje signalConfirmacionSesion(String sesionId) {

		GestorContenido gc=new GestorContenido(session);

		Integer path=null;
		path=gc.obtenerReferenciaContenido(sesionId);

		Transaction tx=session.beginTransaction();

		Sesion s=(Sesion) session.get(Sesion.class,
Integer.valueOf(sesionId));
		s.estado=Konstants.ESTADO_SESION_CONFIRMADA;
		s.pathContenido=path;
		session.update(s);
		tx.commit();


			Mensaje msg=new Mensaje("OK","Confirmed",sesionId);
			return msg;


	}
```

ILUSTRACIÓN 74 EJEMPLO DE MANEJO DE OBJETOS EN HIBERNATE

3.7 CONCLUSIÓN

En esta sección se ha descrito como se han desarrollado, configurado y desplegado los componentes de la aplicación. En general, el esfuerzo de codificación no es muy grande comparado con las complicaciones que lleva consigo la configuración de un sistema distribuido. Si el diseño de una aplicación está bien planteado, la codificación se resume en seguir las guías de los diagramas de actividad y de secuencia. La mayor complejidad en la codificación se encuentra en las imposiciones del sistema distribuido tales como el manejo de errores de la red o del sistema de ficheros sobre el que se esté trabajando.

En el siguiente capítulo se describe las pruebas que se han hecho una vez que se han desarrollado las interfaces de comunicación y acceso a ficheros y base de datos.

4 PRUEBAS Y VALIDACIÓN

4.1 PRUEBAS DE INTERFACES COMMON CARTRIDGE Y WEBSERVICES

Después del desarrollo de la interfaz de intercambio de datos Common Cartridge y presentar los servicios de la extensión de ILLLab en Web Services fue necesario realizar pruebas de comunicación y de conformidad de datos.

4.1.1 PRUEBAS DE INTERFAZ COMMON CARTRIDGE

Las pruebas de interfaz CommonCartridge se realizan mediante un programa en Java que simplemente comprueba que el contenido en una carpeta se liste correctamente en pantalla. Esto lo debe realizar analizando el fichero imsmanifest.xml que especifica Common Cartridge y luego parseando los ficheros HTML que son parte del contenido de los ejercicios de la aplicación ILLLab.

CommonCartridge

Este estándar es uno de tres estándares de servicios (Digital Learning Services) para dar soporte a tecnologías de nueva generación para la educación. CommonCartridge permite exportar e importar ficheros junto con la descripción de los datos para su posterior procesado.

La estructura de ficheros en las que se organizan los datos de la descripción del contenido tiene un esquema definido. El fichero principal es el imsmanifest.xml que incluye una descripción de cómo se encuentra distribuido el contenido en elementos llamados Resources. Cada Resource incluye el tipo (WebLink o WebContent ó LTI) y el nombre del fichero que incluye el contenido junto con ficheros auxiliares como imágenes, iconos, etc. tal como se muestra en las siguiente figuras.

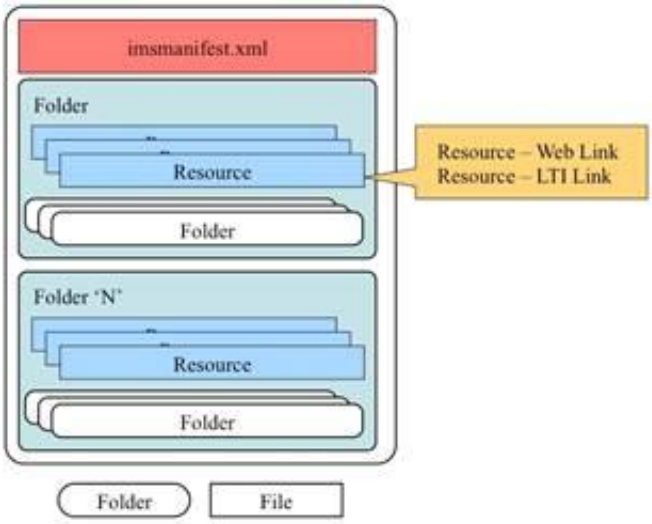


ILUSTRACIÓN 75 ESTRUCTURA DE FICHEROS SEGUN COMMON CARTRIDGE[30]

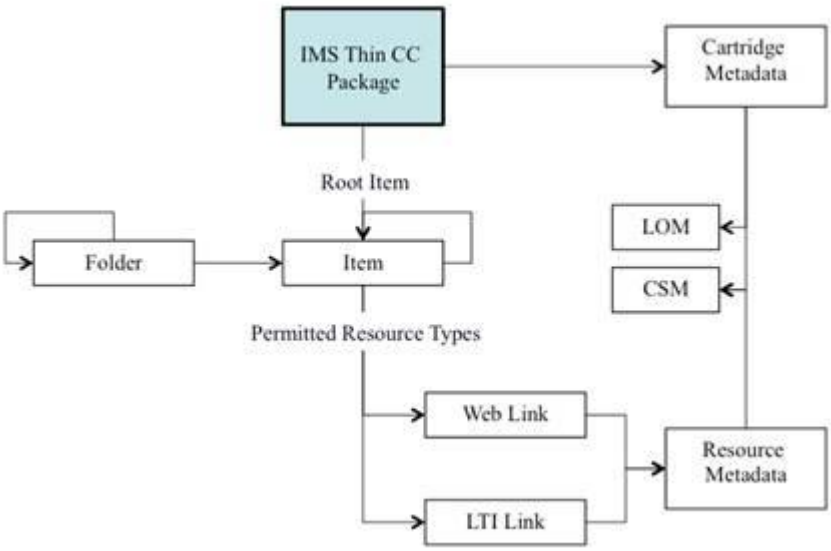


ILUSTRACIÓN 76 TIPOS DE CONTENIDO

LTI Link es un formato que incluye el path a un fichero y es el que se utiliza en este proyecto.

```
<resource identifier="I_00010_R" type="imsbasiclti_xmlv1p0">
              <file href="I_00001_R/BasicLTI.xml"/>
       </resource>
```

ILUSTRACIÓN 77 LTI LINK EN IMSMANIFEST.XML

WebContent apunta a una página estática:

```
<resource identifier="RES-eXetiple_opts44f81a2d209724fd641a" type="webcontent" href="index.html">
  <file href="index.html"/>
  <file href="base.css"/>
  <file href="content.css"/>
```

ILUSTRACIÓN 78 WEB CONTENT EN IMSMANIFEST.XML

El contenido de la extensión de ILLLab se exporta en forma de WebContent debido a que la información de los cuestionarios se edita como página web y se exporta en HTML.

En este proyecto se implementó una interfaz que comprende tres objetos para abstraer la parte de interactuar con CommonCartridge: (1) un parser IMS Manifest, (2) un parser HTML y (3) un objeto que obtiene el contenido para la aplicación.

En las pruebas simplemente se intenta obtener todo el contenido de una partida: la pregunta, las respuestas y un indicador de cual es la respuesta correcta. Estas funcionalidades se programaron en una clase Java que mostraba por pantalla el contenido. La clase se llama MainIMSParser y el resultado de una ejecución es el siguiente:

```
(Pregunta)Joe stepped onto the aeroplane and was met by one
of the cabin crew who showed him to his seat. This was his
first flight and he was feeling quite nervous. His hands
were_____slightly and he was breathing deeply. He walked
along the aisle of the plane and found his seat
```

ILUSTRACIÓN 80 RESULTADO DE OBTENER LA PREGUNTA DE UN EJERCICIO

```
(Respuestas e indicadores)
moving
NOK
trembling
OK
jumping
NOK
```

ILUSTRACIÓN 79 RESULTADO DE OBTENER LAS RESPUESTAS A UN EJERCICIO Y SUS RESULTADOS

Para ejecutar las pruebas es necesario utilizar la herramienta software Eclipse que es la misma que se utiliza para programar la aplicación de ILLLab. Del contenido digital presentado en este proyecto se debe descargar el proyecto Java llamado "CommonCartridgeRestService" al disco duro del PC donde se esté trabajando e importarlo con Eclipse.

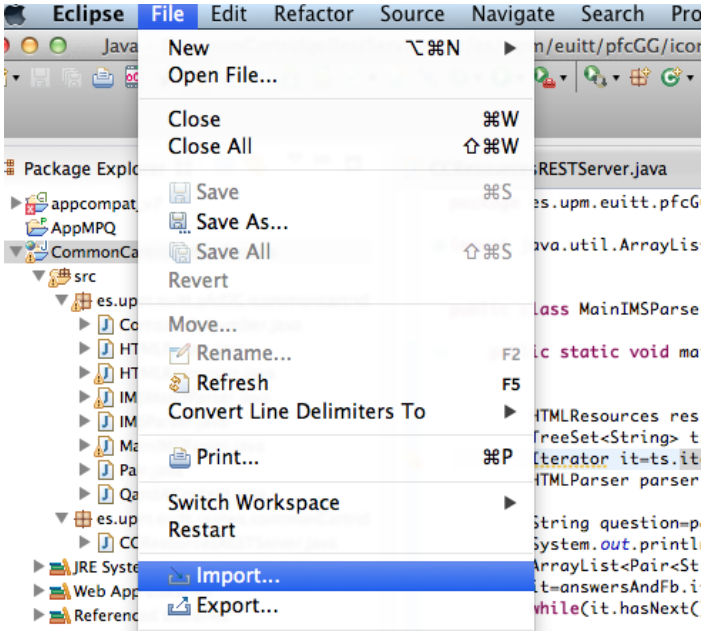


ILUSTRACIÓN 81 IMPORTAR PROJECTO DE PRUEBA PASO 1

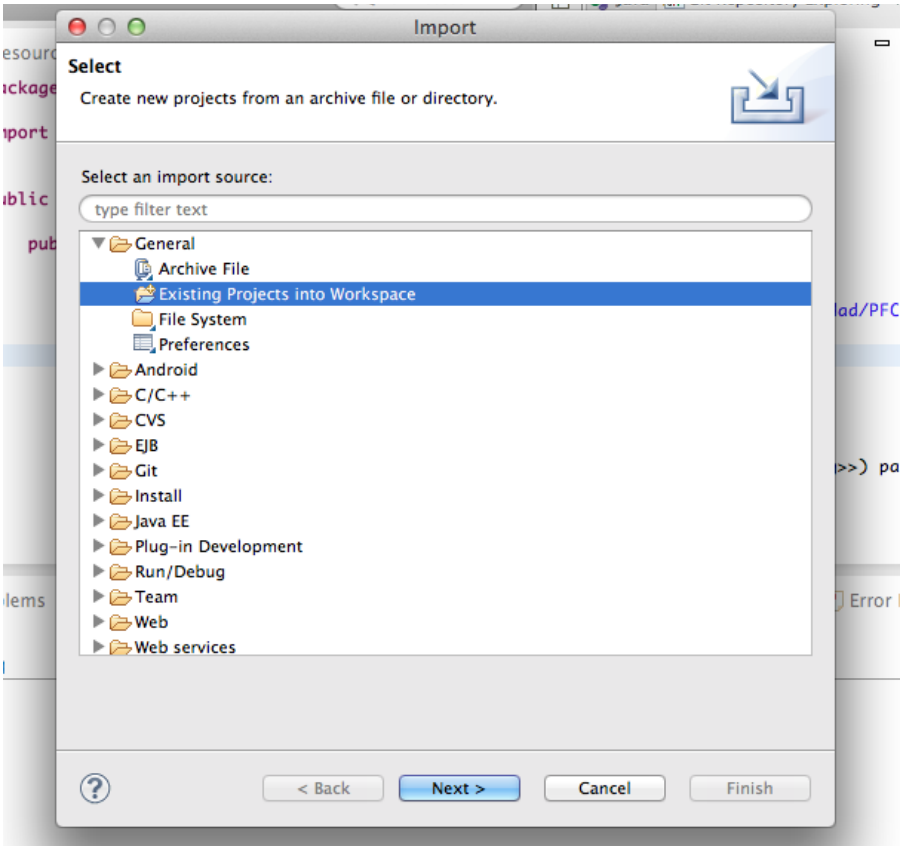


ILUSTRACIÓN 82 IMPORTAR PROJECTO DE PRUEBA PASO 2

Se siguen las instrucciones para importar el proyecto y se ejecuta como proyecto Java haciendo clic derecho sobre la clase que se desea ejecutar, en este caso, MainIMSParser.java.

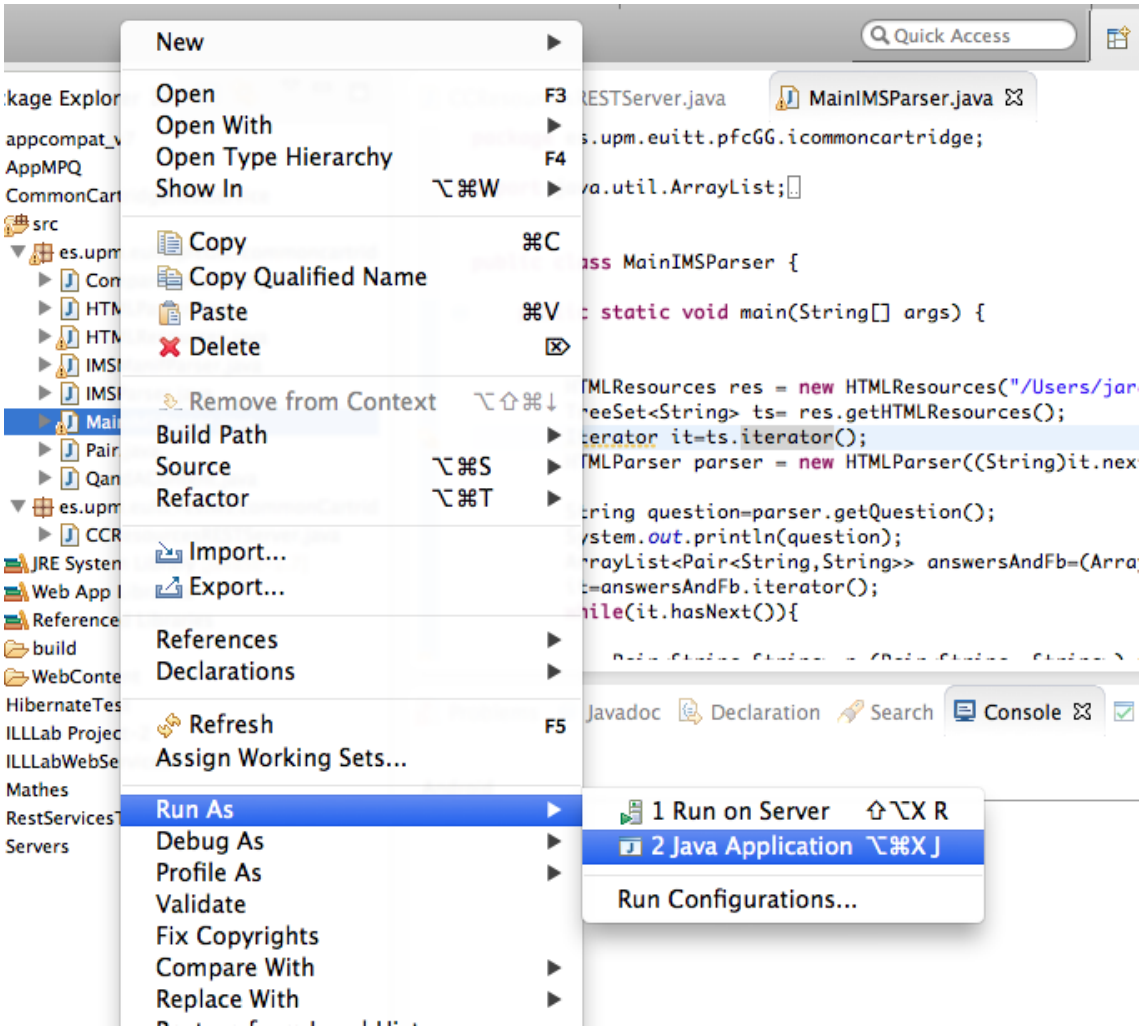


ILUSTRACIÓN 83 EJECUTANDO EL PROGRAMA DE PRUEBA

4.1.2 PRUEBAS DE COMUNICACIÓN CON WEB SERVICE

Hay dos pruebas de Web Services en el proyecto que consisten en la prueba de acceso a los servicios Common Cartridge y en el acceso a los servicios de la extensión de ILLLab. La primera prueba consta de la distribución de la pruebas realizadas anteriormente en un cliente y un Web Service de Common Cartridge por lo que el resultado es el mismo de la prueba anterior. La diferencia es que hay que desplegar el Web Service en un servidor Tomcat que ofrece Eclipse para el depurado de la aplicación.

El despliegue del Web Service Common Cartridge se realiza en dos pasos: (1) importar los proyectos RestServicesTest y CommonCartridgeRestService en el workspace de Eclipse y (2) desplegar el Web

Service haciendo clic derecho en el proyecto CommonCartridgeRestService, eligiendo la opción "Run as" y luego "Run on Server". Después de haber realizados los dos pasos se procede a ejecutar la clase Java Main.java en el proyecto RestServicesTest. El resultado de la prueba debe ser el que se muestra en la siguiente imagen. La prueba muestra un texto con una palabra sin completar y debajo se muestran dos opciones.

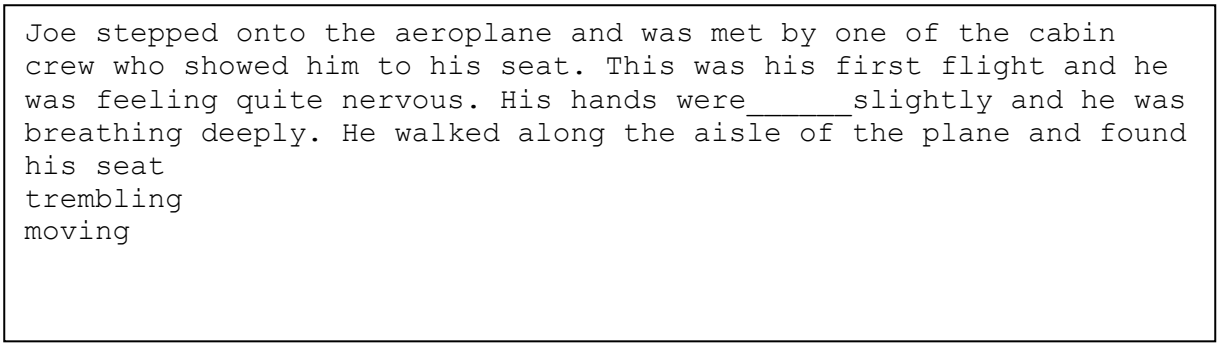


ILUSTRACIÓN 84 RESULTADO DE LA PRUEBA DEL WEB SERVICE COMMON CARTRIDGE

La siguiente prueba es la del Web Service de ILLLab. Esta prueba fue bastante extensa debido a que aquí se incluyeron las pruebas de Hibernate con el objetivo de solucionar los errores de la lógica de negocio como los errores de configuración de Hibernate.

Para realizar esta prueba se debe desplegar el Web Service tal como se ha desplegado el proyecto de la prueba anterior de Common Cartridge. Luego se ejecuta una clase Java que hace de cliente del Web Service. La clase se encuentra en el proyecto RestServicesTest y se llama ILLLabWSTests. Esta clase incluye varias llamadas a los Web Services de la aplicación.

4.2 VALIDACIÓN DE REQUERIMIENTOS

Una vez realizadas las pruebas de integración y de asegurarse del correcto funcionamiento de la lógica de la aplicación, se procede a realizar pruebas de validación de requerimientos utilizando la aplicación simulando el uso que se haría en un entorno real. Estas pruebas tienen como objetivo comprobar que los requerimientos se cumplen y si no es así explicar por qué y cómo mejorar las prestaciones de la aplicación.

Se han utilizado dos tipos de plataformas móviles, la primera un emulador de dispositivo móvil Android y la segunda en un dispositivo real. Para realizar las pruebas con el emulador, el Android SDK pone a disposición del usuario de Eclipse el Android Virtual Device Manager (AVD) que una extensión software para la creación y gestión de emuladores con el que es posible tener una versión virtual de un dispositivo

que funciona con Android. Una vez creado el dispositivo virtual se puede desplegar la aplicación que se haya desarrollado en el mismo.

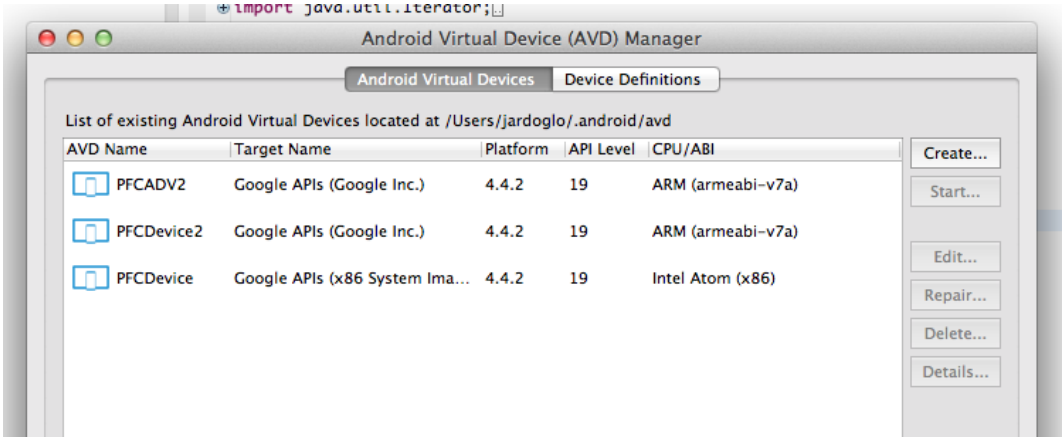


ILUSTRACIÓN 85 ANDROID VIRTUAL DEVICE MANAGER

Las pruebas con el dispositivo Android real se realizan simplemente conectando el móvil al portátil vía un puerto USB y ejecutando la aplicación directamente sobre el móvil. Eclipse detecta el dispositivo automáticamente pero es necesario activar las opciones de desarrollador en el móvil.

Para la validación, tal como se ha explicado, es necesario simular un entorno real por lo que las pruebas han incluido el despliegue de dos emuladores para simular el juego entre dos jugadores, luego se ha desplegado la aplicación en un móvil para observar el funcionamiento en el dispositivo real y finalmente se han hecho pruebas entre un emulador y el dispositivo móvil real para simular el juego entre dos jugadores en un entorno más real que en el que se utilizan dos emuladores. En general, la respuesta de la aplicación en un dispositivo móvil es mejor que en un emulador porque el dispositivo real tiene sus propios procesadores de vídeo y datos mientras que el emulador debe ocupar memoria de un PC o portátil y por lo tanto depende del grado de utilización de memoria y procesado del ordenador en ese momento y las capacidades con las que viene de fábrica.

ILUSTRACIÓN 86 APLICACION ILLLAB EN EMULADOR

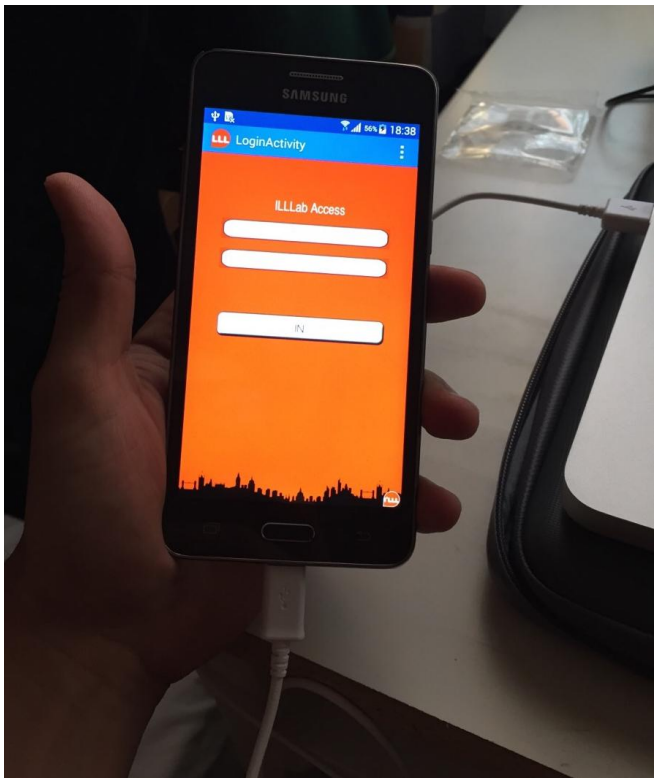


ILUSTRACIÓN 87 APLICACION ILLLAB EN SAMSUNG GRAND PRIME

Luego de hacer las pruebas de funcionamiento se han hecho modificaciones y finalmente se ha reflejado el cambio en las listas de requerimientos.

Lista de requerimientos y validación

Requerimientos funcionales

TABLA 27 TABLA DE VALIDACION DE REQUERIMIENTOS FUNCIONALES

[nº-rf-codigo]	Descripción	Verificación
1-rf-gui	El interfaz gráfico debe permitir elegir entre único jugador y dos jugadores	OK. Se han incluido dos modalidades de juego
2-rf-gui	El usuario puede elegir el tema sobre el que se va a evaluar: gramática, vocabulario o uso del inglés	NOK: Este tipo de ejercicios sólo admite "uso del inglés"
3-rf-gui	El usuario puede empezar con la última partida guardada o con otra. Sólo se guarda la última partida en modo un solo jugador.	NOK: No es posible realizar estas acción debido a que se debe evaluar al alumno realizando una partida entera. A cambio, si se sale de la partida se guarda esa puntuación y el jugador tiene la posibilidad de jugar otra partida
4-rf-dif	El juego no ofrece distintos niveles de dificultad	OK: Este juego es sólo para nivel B2 de ingles
5-rf-gestion	El juego permite borrar una partida previamente empezada	NOK: No se debe dar la posibilidad de gestionar las partidas a los jugadores. Esto dificulta la gestión del sistema y en caso de que sea una partida de dos jugadores
6-rf-puntuacion	El juego permite visualizar la puntuación en un registro de partidas	OK: Hay un listado de partidas terminadas en el cual se puede verificar la puntuación de cada partida
7-rf-	La aplicación sólo	NOK: Hay listados de partidas

	muestra el listado de partidas terminadas	terminadas y activas. Se debe mostrar las partidas activas para interactuar con otro jugador

Requerimientos No Funcionales

TABLA 28 TABLA DE VALIDACION DE REQUERIMIENTOS NO FUNCIONALES

[nº-rnf-codigo]	Descripción	Validación	Verificación
1-rnf-red	La aplicación debe recuperarse de fallos por inconsistencias de la red	Si	Se ha hecho un control de errores por fallo de la red mediante excepciones.
2-rnf-SO	La aplicación debe ser para el sistema operativo Android y el código debe ser reutilizable	No	-
3-rnf-Pantalla	El tamaño de pantalla del dispositivo móvil debe ser tenido en cuenta	Si	Prueba de adaptabilidad de la aplicación. Nivel de adaptabilidad: bajo o alto. Se adapta o no se adapta.
4-rnf-base de datos	La base de datos debe permitir el fácil añadido de datos	Si	Facilidad al añadir datos a la aplicación mediante el path donde se encuentra el contenido
5-rnf	La aplicación debe ser compatible con el formato estándar Common Cartridge	Si	Se ha realizado una interfaz para leer el fichero imsmanifest.xml y el contenido. Esto esta definido en las clases de parseo de los ficheros
6-rnf	Se debe tener en cuenta la movilidad de los usuarios y que los	Si	La coordinación en la generación y juego de una partida entre dos

	eventos ocurren asíncronamente		jugadores se realiza bajo control. No se observan errores
7-rnf	El contenido y la gestión de usuarios se realiza dentro de la red de la institución, no en el dispositivo móvil	-	La infraestructura está pensada para el caso definido en este requerimiento por lo que debe haber un administrador del sistema en cada institución que desee utilizar la aplicación.

Una vez comprobados los requerimientos de la aplicación se puede decir que se ha terminado el trabajo de desarrollo. Sin embargo hay un aspecto importante a tener en cuenta en los proyectos de ingeniería que son lo cálculos de cuánto cuesta realizar un proyecto y en cuánto tiempo se puede amortizar por lo que el siguiente capítulo se destina completamente a esta actividad.

4.3 CONCLUSIÓN

En este capítulo se ha descrito cómo se ha realizado la fase de pruebas. Ha habido pruebas a distintos niveles: (1) a nivel de integración de componentes y (2) a nivel de cumplimiento de funcionalidades siendo (1) un nivel más técnico que (2) debido a que se corrigen errores de configuración de las tecnologías que se utilizan tales como mysql, Hibernate, el estándar Common Cartridge, etc.

Las pruebas en (1) se han realizado sistemáticamente hasta liberar de errores cada capa de software. Después de haber liberado de errores de configuración se ha pasado a corregir la lógica de negocio que se programó en el servicio web y en las actividades que controlan el funcionamiento de la parte móvil de la aplicación. Parte de los requerimientos que se deben cumplir aquí se reflejan en la tabla de requerimientos no funcionales.

Finalmente, una vez que los datos se muestran correctamente en la aplicación se pone a prueba el uso de la aplicación contrastando las funcionalidades implementadas con la tabla de requerimientos funcionales donde se indican todos los requisitos que se deben cumplir. En caso de que un requerimiento no se pueda cumplir se modifica la aplicación ó se indica por qué no se puede cumplir.

En general, los resultados de integración a nivel técnicos fueron los más difíciles de solucionar debido a que es una tarea muy técnica en la que hay que prestar especial atención a cada especificación en MySQL y Hibernate. El diseño de la base de datos en este caso no es algo crucial en el funcionamiento pero las

imposiciones de las bases de datos relacionales deben ser tenidas en cuenta para garantizar la escalabilidad e integridad de los datos.

Por último, una vez que se completan las funcionalidades que se deben cumplir y las correcciones en el funcionamiento es necesario documentar el código, o al menos dejar comentarios para generar los correspondientes JavaDoc para introducirlos en la especificación de los métodos de la aplicación.

5 PLANIFICACIÓN

5.1 ESFUERZO Y PRESUPUESTO

En los apartados siguientes se incluyen los aspectos relativos al presupuesto y la planificación temporal del proyecto.

Lo primero que se va a ver es una Estructura de Descomposición del Proyecto, o EDP. La EDP sirve para dividir el proyecto en elementos de trabajo. Tendrá una estructura tipo árbol para facilitar la comprensión de las distintas fases.

Una vez vista la EDP, se ha hecho una estimación de la duración del proyecto. Habitualmente, en los proyectos de software, se hace esta estimación previamente al desarrollo. Existen varios métodos aceptados, pero para este proyecto se ha optado por el método COCOMO[31].

Para la planificación temporal del proyecto se ha incluido un cuadro con todas las tareas previstas en el proyecto, su división en paquetes individuales de trabajo y una previsión de la duración en horas de cada uno. Con ello será posible contabilizar las horas previstas para el desarrollo del proyecto, indispensables para evaluar el coste de dicho aspecto en el global del proyecto. Posteriormente se han ordenado esas tareas en un diagrama de Gantt, que muestra todos los paquetes de trabajo en un despliegue por meses.

El último punto será el presupuesto estimado del proyecto, incluyendo una posible estimación de rentabilidad del mismo.

5.1.1 EDP

Al tratarse de un videojuego, y en definitiva de un proyecto de software, la EDP o Estructura de Descomposición del Proyecto será la habitual en este tipo de proyectos.

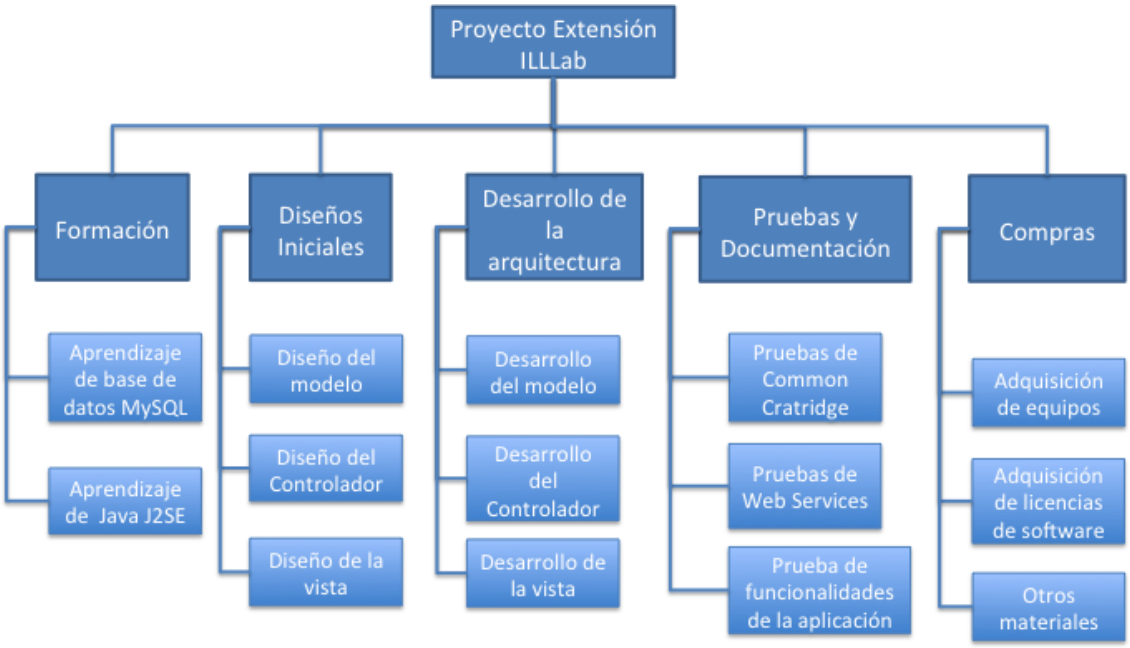


ILUSTRACIÓN 88 DIAGRAMA EDP DE PROYECTO ILLLAB

5.1.2 ESTIMACIÓN PREVIA.

Aunque esta memoria se ha realizado a la conclusión del proyecto y es conocido ya con exactitud el tiempo empleado en él, por su interés se verá a continuación una estimación de la duración de un proyecto de software de las dimensiones de éste.

Es habitual realizar esta estimación antes de acometer cualquier proyecto. Se empleará el método COCOMO de estimación de carga de trabajo y duración del proyecto.

El programa va a constar de 5737 líneas de código. El Modo de desarrollo del software es en consecuencia ORGÁNICO.

Las ecuaciones que se usan son:

$$MH_{nom} = 3.2(KLD)^{1.05} \qquad (1)$$

Donde KLD son las líneas de código en miles y

$$MH = MH_{nom} \prod_{i=1}^{1=15} (EM)_i \qquad (2)$$

donde EM son los multiplicadores de esfuerzo.

Los EM de la aplicación son:

TABLA 29 TABLA DE EM [32]

RELY	0.75
DATA	0.94
CPLX	0.85
TIME	1
STOR	1.21
VIRT	1.30
TURN	1.15
ACAP	0.75
AEXP	1.29
PCAP	0.70
VEXP	1
LEXP	1
MODP	1
TOOL	1
SCED	1

Dentro de los multiplicadores se ha considerado que la formación de los programadores no era muy alta, para tratar de introducir en la estimación la fase de aprendizaje.

Los resultados con esos multiplicadores son:

$$MH_{nom} = 20.03$$

$$MH = 14.92$$

El tiempo de desarrollo estimado será :

$$T_{des} = 2.5(MH)^{0.36} = 6.61\ meses$$

5.1.3 PLANIFICACIÓN DEL PROYECTO.

Se ha dividido la planificación del proyecto en tareas. Dentro de cada tarea, se subdivide la misma en paquetes de trabajo individuales.

Se ha optado por poner la formación como una de las tareas presentes en este proyecto. Como es lógico, esta tarea está presente debido a que se trata de un primer proyecto con algunos detalles técnicos que requieren conocimientos especiales.

Como el interés del proyecto radicaba en ver el proceso íntegro de creación, utilizando un único empleado, ha parecido lógico incluir esta tarea en la planificación ya que ha sido una de las más exigentes en cuanto a consumo de tiempo y esfuerzo.

Los diseños previos son fundamentales en el desarrollo y cuanto más completos sean los diseños previos y más concienzudos los análisis de los distintos problemas no sólo no se pierde tiempo sino que a la larga se ahorra, puesto que se evita el tener que rehacer el trabajo muy a menudo.

Se incluye en esta parte el desarrollo de todos los modelos presentes en el juego.

Una vez se tienen los diseños hechos, se pasa al desarrollo de la arquitectura. Aquí se incluyen desde los diagramas de clases principales hasta la implementación del código de la aplicación.

Se integrarán en esta tarea toda la parte gráfica anteriormente creada con el entorno físico, y se añadirán los efectos multimedia al juego.

Finalmente, se tienen las tareas de pruebas y de documentación. Las pruebas ocupan gran parte del desarrollo, puesto que es crucial que el rendimiento del juego sea bueno, tanto en el emulador como en el teléfono.

La documentación, como es esta memoria, resulta fundamental de cara a trabajos futuros, como elemento de consulta.

También se ha creado una tarea adicional, compras, que lógicamente será previa al desarrollo, en donde se adquirirán todos los equipos y material necesarios para desarrollar el juego.

Resumiendo, se debe ocupar un total de un mes en el trabajo de comunicación y mock-up. Luego se pasa al diseño donde se debe reservar un tiempo de un mes para el acabado y la comprobación de las funcionalidades con la lista de requerimientos. Luego se pasa al desarrollo que requiere un esfuerzo de unos 3 meses y finalmente las tareas de pruebas y documentación que ocuparían aproximadamente dos meses. Entre las tareas de comunicación y desarrollo se debe hacer un estudio previo de las tecnologías que pueden soportar la arquitectura para tener en cuenta en la fase de compras de equipos y licencias software.

5.1.4 DIAGRAMA DE GANTT.

Se incluye a continuación un diagrama temporal con un desglose por meses de los distintos paquetes de trabajo.

Se trata de un diagrama muy simple, con una evolución en serie de las distintas tareas. Hay que tener en cuenta que este proyecto lo lleva a cabo una sola persona, con lo que llevar tareas en paralelo es posible sólo para el asesoramiento de tecnologías y las compras que son parte de la gestión.

ILUSTRACIÓN 89 DIAGRMA GANTT PROYECTO ILLLAB

El resultado final es de 1.008 horas, lo que supone, tomando 20 días laborables al mes de 8 horas cada uno, aproximadamente 6,6 meses, que es muy similar al valor calculado en la estimación COCOMO.

5.1.5 PRESUPUESTO.

En todo proyecto de software como éste, existen básicamente tres grandes categorías que suponen la mayor parte del coste:

- Coste de Materiales.
- Coste de Licencias.
- Coste de Mano de Obra.

5.1.5.1 COSTE DE MATERIALES.

En este apartado se engloba el precio del uso de los diversos equipos empleados para desarrollar el presente trabajo, describiendo tanto el precio del hardware como el del software. También se añadirá un resumen del conjunto de material de oficina utilizado durante la realización del proyecto.

➢ *Recursos Hardware.*

Se ha considerado que un PC se amortiza en unos 3 años. El periodo de utilización del mismo será toda la duración del proyecto, los 15 meses calculados en el apartado anterior.

Finalmente, se ha añadido como recurso una impresora láser, con un periodo de amortización de 4 años. En principio se utilizará para la fase de documentación, y en algún uso ocasional a lo largo del proyecto. En total se ha estimado un uso de 5 horas.

TABLA 30 TABLA DE COSTE DE MATERIALES

Hardware	Precio	Amortización	Tiempo utilizado	Total en €
PC	1.200 €	3 años	7 meses	233 €
Impresora Láser	200 €	4 años	5 horas	0,13 €
Teléfono Móvil LG	200 €	2 años	6 meses	50 €
TOTAL RECURSOS HARDWARE				283,46 €

➢ *Recursos Software.*

En recursos software se incluye el coste de las licencias de todos los programas comerciales que se han utilizado. Como es habitual en el software, se han considerado periodos de amortización de entre 3 y 5 años.

TABLA 31 TABLA DE COSTES DE RECURSOS SOFTWARE

Software	Precio	Tiempo utilizado	Total en €
Office 365	10€/mes	7 meses	70 €
Paint	0 €	3 meses	0 €
TopCoder	0 €	3 meses	0 €
Eclipse	0 €	4 meses	0 €

MySQL Workbench	0 €	4 meses	0 €
Adobe Photoshop	12,09 €/mes	1 mes	12,09 €
TOTAL RECURSOS SOFTWARE			82,09 €

➢ *Material de Oficina.*

En material de oficina se tiene en consideración todo el papel, artículos para imprimir y costes de encuadernación del proyecto.

TABLA 32 TABLA DE COSTES DE MATERIAL DE OFICINA

Material	Total en €
Papel	30 €
Tóner	200 €
Encuadernación	70 €
Lápices, bolígrafos, cuadernos…	20 €
TOTAL MATERIAL DE OFICINA	320 €

Llegados a este punto, se puede realizar el cálculo final del coste que supone el conjunto de recursos utilizados:

TABLA 33 TOTAL DE COSTES DE MATERIALES

Descripción	Total
Total recursos hardware	283,46 €
Total recursos software	82 €
Total materiales de oficina	320 €
TOTAL MATERIALES	685,55 €

5.1.5.2 COSTE DE MANO DE OBRA.

Los costes que se incluyen en este apartado, se derivan del pago por la mano de obra utilizada para realizar el presente proyecto. En este tipo de costes aparecen los derivados del diseño de Ingeniería, redacción y realización del libro (mecanografía).

Se ha preferido desglosar el proyecto en estas dos atribuciones, debido que aunque todo ha sido realizado por la misma persona no puede considerarse el mismo coste/hora a cada tarea.

Este proyecto es de más o menos 7 meses incluyendo festivos y fines de semana que cuentan como no laborables y que en total hacen alrededor de 1.008 horas. Suponiendo que toda la labor de ingeniería la lleva a cabo un ingeniero junior, se puede suponer un coste por hora de 12,5 €/h, que aproximadamente correspondería a unos 25.000 € al año.

TABLA 34 TABLA DE COSTES DE MANO DE OBRA

Realización	Coste por hora	Total horas	Total en €
Diseño de Ingeniería y redacción del Proyecto	12,50 €	800	10000
Realización del libro	12,50 €	208	2.600 €
TOTAL MANO DE OBRA			12.600

5.1.5.3 PRESUPUESTO DE EJECUCIÓN MATERIAL

Se calcula como la suma del coste total material y el coste total debido a la mano de obra.

TABLA 35 PRESUPUESTO DE EJECUCION MATERIAL

Descripción	Total
Coste total del material empleado	685,55 €
Coste total de la mano de obra	12.600 €
PRESUPUESTO DE EJECUCIÓN MATERIAL	13.285,55 €

Aunque no es éste un proyecto clásico de ingeniería se ha optado por incluir el presupuesto de ejecución material.

5.1.5.4 PRESUPUESTO DE EJECUCIÓN POR CONTRATA.

Se incluyen en este apartado los gastos derivados del uso de las instalaciones donde se ha llevado a cabo el proyecto, cargas fiscales, gastos financieros, tasas administrativas y derivados de las obligaciones de control del proyecto.

De igual forma, se incluye el beneficio industrial y los posibles imprevistos. Para cubrir estos gastos se establece un recargo del 22% sobre el importe del presupuesto de ejecución material.

Este proyecto no se ha llevado a cabo en ninguna empresa, pero se ha considerado más riguroso incluir esta partida en el presupuesto total.

TABLA 36 PRESUPUESTO DE EJECUCION POR CONTRATA

Descripción	Total
Coste total de ejecución material	13.285,55 €
22% de gastos financieros, beneficios, etc.	2.922,82 €
PRESUPUESTO DE EJECUCIÓN POR CONTRATA	16.208,38 €

5.1.5.5 HONORARIOS FACULTATIVOS.

Al no tratarse de un proyecto clásico de ingeniería no procede incluir los honorarios facultativos ya que no es necesario visar ningún documento.

5.1.5.6 PRESUPUESTO TOTAL.

El coste total del proyecto es la suma del presupuesto de ejecución por contrata, y los honorarios facultativos (nulo en este caso). Habrá que añadir al montante total un 16 % de IVA.

TABLA 37 PRESUPUESTO TOTAL

Descripción	Total
Presupuesto de ejecución por contrata	16.208,38 €
Honorarios facultativos	0 €

Presupuesto total	16.208,38 €
16 % IVA	2.593,34 €
PRESUPUESTO FINAL	18.801,72 €

5.1.5.7 ANÁLISIS DEL PRESUPUESTO.

Como se ve el coste final del producto es de 18.801,72,05 €. Hay que tener en cuenta que se ha incluido en el cómputo de horas un largo proceso de formación en las herramientas utilizadas, así como licencias de software de primer nivel que lógicamente van a encarecer el producto.

El presupuesto se ha realizado teniendo en cuenta que una única persona lleva a cabo toda la creación del producto, pero existen otras alternativas.

Para empezar, un ingeniero puede encargarse de todo el diseño para dejar que sea un técnico en informática quien se encargue de implementar el código. De igual forma, suelen ser artistas gráficos quienes se encargan de la parte visual, sin ser necesario tenerles en plantilla puesto que cada vez es más habitual que los diseñadores gráficos ofrezcan sus servicios de forma freelance a través de Internet. Sólo habría que comprar o encargar los modelos que hicieran falta.

Lo mismo pasaría con aquellos efectos de sonido o melodías que se quieran utilizar.

Como alternativas a las licencias adquiridas, existe software de modelado gratuito que para el nivel de dificultad de los gráficos que se usan en este trabajo podrían servir.

Por último, hay que comentar que para posibles futuros proyectos los gastos van a ser mucho más reducidos. Para empezar, la formación no hay que repetirla, puesto que básicamente se utilizarán los mismos conceptos aprendidos en este proyecto. En cuanto a la parte de desarrollo, gran parte del código es perfectamente aprovechable para otros juegos, al menos los que sean de un estilo parecido.

5.2 EXPECTATIVAS DE BENEFICIO.

Lo más interesante dentro del desarrollo de los juegos realizados para Android es que existe una posibilidad muy real de llegar a comercializarlos.

Modelo de negocio de las aplicaciones (apps)

Antes de empezar a pensar en cómo obtener beneficio de un juego, es imprescindible comprender como funciona el modelo de negocio de las apps para dispositivos móviles.

- Aplicación de pago

El usuario descarga la aplicación en su dispositivo previo pago del precio establecido por el desarrollador. Dependiendo de la plataforma, las tiendas de aplicaciones fijan unos rangos de precios que los desarrolladores seleccionan como precio de venta al público. En el caso de Android, al desarrollador le corresponde el 70% de la venta de la aplicación

- Aplicación gratuita

El usuario descarga la aplicación sin abonar nada de dinero. Este modelo de negocio es muy habitual por la escasa predisposición de algunos usuarios de Internet en descargar contenido de pago. Un dato muy curioso es que en la tienda de aplicaciones de Android (Google Play) cerca del 90% de las aplicaciones que se descargan son gratuitas.

La monetización de este tipo de aplicaciones suele ser mediante la inserción de publicidad, suele ser un banner situado en la parte superior o inferior de la pantalla. La desventaja de este tipo de aplicaciones móviles es que estos anuncios suelen resultar muy molestos para el usuario, ya que dificultan la navegación y ralentizan el funcionamiento de la app.

- Aplicación gratuita con contenido de pago adicional

Un modelo de negocio que se está extendiendo con más potencia es un híbrido entre las dos opciones anteriores. El usuario descarga la aplicación sin desembolsar ninguna cantidad de dinero, pero tiene acceso limitado al contenido de la aplicación.

La monetización puede buscarse a través de dos métodos simultáneos: ingresos generados por el pago para desbloquear el contenido adicional (compra de contenidos desde la aplicación o In-App Purchase) y por las impresiones generadas por publicidad. En este caso, en el momento en que un usuario efectúe una compra, se deberán desactivar los anuncios en su dispositivo.

En definitiva, la aplicación para el dispositivo móvil debe ser gratuita, debido a que es sólo una parte de la infraestructura de gestión de contenido.

¿Cómo vender el LMS + app?

Una vez vista la manera de vender una aplicación, lo importante es vender la infraestructura entera, es decir, la instalación de la infraestructura con el manual de usuario, etc. Esta infraestructura se entiende como una plataforma LMS que permite gestionar el contenido y los datos para la aplicación y lo que la aplicación genera. Hay dos posibilidades, mantenerla y facturar bajo un esquema de cobro y otra es vender la infraestructura para que la mantenga una empresa que se dedique a la enseñanza.

Asumiendo que una institución que se dedique a la educación no quiere asignar recursos para mantener una infraestructura, he aquí un desglose de cómo los proveedores de la mayoría de los LMS cobran por su software.

- Pago por estudiante

Este es el modelo de precios más común en el que se paga una tarifa fija por alumno (independientemente de la cantidad de capacitación que están recibiendo). Además, a menudo hay una cuota de instalación de una sola vez.

El rango de precios es de alrededor de US$ 5 por usuario por mes, pero los precios pueden bajar a medida que aumenta el volumen a unos US$ 0.50 por usuario por mes para grandes empresas con muchos alumnos.

LMS con este modelo de precios:

  - SumTotal Systems
  - Skillsoft
  - Taleo
  - Latitude Learning
  - Edvance360

- Pago por uso

Esto puede significar cosas diferentes dependiendo del proveedor de LMS, por lo que hay muchas opciones: cobrar una tarifa por usuario por módulo ó un cargo por curso por usuario (esto es muy común), una tarifa basada en elementos o materiales entregados por curso, o una tarifa basada en el número de asistentes de clase.

Rango de precios: Depende del modelo específico y su volumen, pero se espera que entre US$ 0.50- US$ 10 por alumno por curso.

LMS con este modelo de precios:

  - SuccessFactors
  - Cornerstone OnDemand
  - DigitalChalk

- Derechos De Licencia

Esto es un pago inicial por adelantado que es el costo para acceder al software, o se trata de una cuota para acceder al software por un período específico de tiempo (mensual, anual, etc.). También puede haber una cuota anual de apoyo.

Rango de precios: Menos de US$ 500 a decenas de miles de dólares (por ejemplo, US$ 20,000 anuales).

LMS con este modelo de precios:

- Desire2Learn
- Halogen
- Meridian

- Otros Modelos de precios

Otros modelos de fijación de precios es posible encontrar incluyen ilimitado número de usuarios con cuota fija (ej. Interactyx) y pago por curso (ej. CourseWebs). Estos también podrían combinarse con uno de los modelos anteriores (es decir, pago por curso con un cargo adicional de licencia de una sola vez).

Venta a institución que tenga un LMS integrado

La venta a un cliente que tenga un LMS es un modelo de negocio distinto. Aquí la negociación es simple debido a que el coste final de venta es el coste del desarrollo más un margen de ganancia y habría que agregar el coste de integración al sistema del cliente final. En el contrato debería reflejarse también que el coste de mantenimiento es aparte de la venta de la aplicación.

5.3 CONCLUSIÓN.

Existen dos tipos de clientes, uno es el cliente final que utiliza la aplicación y el sistema LMS y otro es un cliente que ya tiene un LMS en su institución y desea integrar la aplicación en su sistema. Para cada cliente el modelo de negocio es distinto.

En el caso de que un cliente final quiera integrar la aplicación en su LMS entonces se debe plantear un proyecto de integración por lo que es difícil decir qué precio poner a la venta de la aplicación debido a que en este caso sólo una parte del desarrollo vale como producto final.

En el caso de que el cliente final requiera de un LMS y la aplicación el coste debe incluir no sólo la parte del desarrollo sino también la infraestructura que se necesita para manejar un cierto volumen de clientes. De todas formas, haciendo una aproximación preliminar, con el esquema de pago por uso y un cliente final

con una cantidad de unos 200 alumnos, en el que se cobra una mensualidad de alumno por curso de €3 entonces es posible amortizar la suma de €18.801,72 en 31,34 meses que equivale a menos de tres años.

6 CONCLUSIONES

6.1 DESARROLLO DEL PROYECTO Y SUS DISTINTOS ASPECTOS

Elección de la metodología

La elección de la metodología es decisiva para la planificación de todo el proyecto. Si bien parece intuitiva al principio, es necesario saber de antemano qué características tiene el proyecto y las posibilidades que ofrecen los modelos. En este caso la elección del modelo es inmediato porque el desarrollo lo realiza una persona, pero en el caso de tener varios grupos o distintos desarrolladores realizando tareas diferentes la planificación es más determinante.

En conclusión, la base de todo proyecto debe empezar por la definición y la elección de la metodología para tener una visión general de todas las actividades para así poder planificar y evaluar riesgos.

Desarrollo

La fase de comunicación y mock-up se ha realizado de una manera informal mediante entrevistas. No se ha utilizado cuestionarios ni nada parecido. Simplemente se ha hecho una definición textual de lo que debe ser la aplicación y el uso que se le daría.

La parte de Mock-up no ha ayudado de gran manera especialmente en este caso porque el cliente está familiarizado con las tecnologías móviles y gestores de contenido por lo que esa fase ha sido simplemente mostrar la aplicación en la versión de Mock-up y confirmar las funcionalidades.

De todas formas, es recomendable esta práctica en cualquier caso para aclarar cualquier duda dado que siempre hay malentendidos.

Diseño

El diseño se evalúa según en qué medida ha ayudado al desarrollo de la aplicación. En este caso se ha elegido el modelo MVC (Modelo-Vista-Controlador) que es un modelo pre establecido por lo que no hay que inventar otro desde el principio. Es recomendable elegir este tipo de modelos porque aporta una visión global de la aplicación y permite la fácil definición de componentes e interfaces.

Además del modelo simple MVC se han diseñado todos los componentes que integran las tres partes del modelo. En los casos en que el diseño de algunos componentes ya esté desarrollado en alguna librería, es importante expresar la interfaz de cada componente y sus funcionalidades en el diseño. En caso de no hacerlo, se pierde visión a la hora de programar la aplicación. Esto no sólo ocurre para los componentes que están desarrollados sino también para los que se van a desarrollar. El diseño debe incluir todos los aspectos de la aplicación tales como el intercambio de mensajes en los diagramas de flujo y de actividad.

En este proyecto ha faltado el completo desarrollo de la una parte de la arquitectura que es el manejo de fallos. Esto se ha realizado parcialmente debido a que no ha sido propiamente reflejado en los requerimientos por lo que a la hora de realizar la lista de requerimientos es necesario tener una lista de aspectos que se deben cubrir. Esto no es característico de éste proyecto por lo que se puede obtener una lista definida en documentos de buenas prácticas de diseño software.

Desarrollo

La fase de desarrollo ha sido fácil desde el punto de vista de programar las clases que se han generado desde el diseño. La parte más complicada ha sido la de configurar las herramientas de trabajo para poder trabajar con Web Services, Hibernate, MySQL y Android.

Pruebas

Las pruebas se han realizado en dos fases, en la primera se realizaron pruebas de integración de componentes para corregir fallos de programación y de configuración y en la segunda se han realizado pruebas del funcionamiento de la aplicación para comprobar que el flujo de las actividades es lógico y que se cubren las necesidades descritas en los requerimientos.

El único error que ha surgido en esta fase es el de la definición de la aplicación en sí. Ha sido necesario agregar algunas funcionalidades por la falta de definición en la fase de comunicación. Si bien se había realizado un Mock-up para mostrar cómo sería la aplicación, al final las necesidades del cliente pueden cambiar a lo largo del desarrollo del proyecto. Esto es un riesgo que se puede tener en cuenta en la fase de diseño con partes de la aplicación que sean reutilizables. Sin embargo, las modificaciones siempre requieren un esfuerzo extra.

Planificación, presupuesto y potencial beneficio

El cálculo del esfuerzo se ha realizado mediante el método COCOMO que es un método aceptado en el desarrollo de software. La planificación se ha realizado teniendo en cuenta lo que se ha realizado durante el proyecto. Sin embargo, la estimación del tiempo suele no ser exacta debido a los contratiempos que puedan surgir en otras situaciones.

En cuanto a los beneficios que se pueden obtener, hay varios modelos de negocios dependiendo del tipo de cliente al que se intenta vender la aplicación. En este caso se ha tenido en cuenta que se vende la aplicación junto con un sistema de gestión de contenido por lo que el cálculo de amortización se basa en lo realizado durante el desarrollo del proyecto y algunos costes de materiales e impuestos a modo orientativo. Para realizar un cálculo más exhaustivo habría que incluir el coste de comercialización del producto y además ver vías de financiación que es lo que suele mantener a una empresa en la que los plazos de cobro son a veces inciertos o de largos períodos.

6.2 MEJORAS Y FUTURO DE LAS APLICACIONES

Éste proyecto sólo ha introducido una parte innovadora en el aspecto de manejo de datos de la aplicación. De todas formas, estas aplicaciones ya existen por lo que el grado de madurez de ILLLab con respecto a las aplicaciones existentes es bajo no sólo por las prestaciones que puede dar la aplicación sino por la novedad que pueda introducir otro tipo de interacción con el móvil.

Algunos aspectos mejorables de la aplicación están en la parte de manejo de errores, el cambio de la plataforma Android a una aplicación multiplataforma a través de la web o el desarrollo de la versión para iOS. El desarrollo de otro tipo de interacción mediante reconocimiento de voz para la corrección en la pronunciación sería algo que se valoraría más por parte de los usuarios finales. Además, los avances en inteligencia artificial y su aplicación en el desarrollo de aplicaciones móviles podrían mejorar la interacción con el usuario y así nuevas aplicaciones pueden salir al mercado e incluso pueden llegar a cambiar el modo de aprendizaje del inglés.

7 BIBLIOGRAFÍA

[1] Modelo en espiral, http://blog.iedge.eu/wp-content/uploads/2011/09/IEDGE-ciclo-de-vida-desarrollo-software-4.jpg

[2] [5] Unified Modelling Language, http://www.uml.org

[3] Ieee standard glossary of software engineering terminology, 1990. IEEE Std 610.12-1990.

[4] Génova, Valiente, & Marrero, Sobre la diferencia entre análisis y diseño, y por qué es relevante para la transformación de modelos, 2006, http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/Is1/DiferenciaAD.pdf

[6] Herramienta de desarrollo Netbeans, https://netbeans.org

[7] Herramienta de desarrollo Eclipse, https://eclipse.org

[8] Application Programming Interface

[9] Software Development Kit de Android, https://developer.android.com/sdk/index.html

[10] Structured Query Languaje definition, http://www.w3schools.com/sql/sql_intro.asp

[11] Herramienta de gestión de bases de datos SQL, MySQLWorkbench http://www.mysql.com/products/workbench/

[12] Dahn, Ingo Aspect Project, Common Cartridge is not SCORM

[13] Herramienta de gestión de contenidos eXe, http://exelearning.org

[14] Herramienta de diseño UML Topcoder, https://www.topcoder.com/tc?module=Static&d1=dev&d2=umltool&d3=description

[15] Requerimientos de sistema para herramienta MIT App Inventor de Google, http://appinventor.mit.edu/explore/ai2/setup.html

[16] Emulador de móvil Android para MIT, http://appinventor.mit.edu/explore/ai2/mac.html

[17] Descarga de aiInstaller de MIT, http://appinventor.mit.edu/explore/ai2/setup-emulator.html#step2

[18] Actualización Java, https://www.java.com/es/download/

[19] Systems and software engineering-Architecture description, http://cabibbo.dia.uniroma3.it/asw/altrui/iso-iec-ieee-42010-2011.pdf

[20] Sistemas distribuidos, Concepto http://www.monografias.com/trabajos16/sistemas-distribuidos/sistemas-distribuidos.shtml

[21] Concepto de Actividad Android, http://developer.android.com/guide/components/activities.html#Creating

[22] Qué son los Web Services?, https://docs.oracle.com/javaee/6/tutorial/doc/gijvh.html

[23] Lógica de negocio, http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rivera_l_a/capitulo4.pdf

[24] Servidor Apache Tomcat, http://tomcat.apache.org

[25] Tutorial de despliegue de Web Service REST en Eclipse, http://help.eclipse.org/kepler/index.jsp?topic=%2Forg.eclipse.stardust.docs.wst%2Fhtml%2Fwst-integration%2Fdynamic-web-proj.html

[26] Añadir librerías a proyecto Java en Eclipse, http://es.wikihow.com/añadir-un-jar-a-un-proyecto-en-eclipse-(java)

[27] Poner la contraseña root en MySQL, http://dev.mysql.com/doc/refman/5.6/en/resetting-permissions.html

[28] Tutorial de configuración Hibernate en Eclipse, https://kaanmutlu.wordpress.com/2011/07/30/hibernate-installationsetup-on-eclipse-ide/

[29] Configuración específica de Hibernate para ILLLab, https://kaanmutlu.wordpress.com/2011/07/30/hibernate-installationsetup-on-eclipse-ide/

[30] Estructuras de ficheros en Common Cartridge, http://www.imsglobal.org/cc/CCv1p0thin/ims_thinCC_impl-v1p0.html#_Toc419291999

[31] Modelo de costes COCOMO, http://www.sc.ehu.es/jiwdocoj/mmis/cocomo.htm

[32] Definición de EMs de COCOMO, http://sunset.usc.edu/research/COCOMOII/cocomo81_pgm/